NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARDograph No.309

THREE DIMENSIONAL GRID GENERATION FOR COMPLEX CONFIGURATIONS – RECENT PROGRESS

by

J.F.Thompson
Department of Aerospace Engineering
Mississippi State University, Mississippi 39752
USA

and

J.L.Steger
NASA Ames Research Center
Moffett Field, California 94035
USA

Edited by

H.Yoshihara
Boeing Company
Seattle, Washington 98124
USA

This AGARDograph has been produced at the request of the Fluid Dynamics Panel of AGARD.

THE MISSION OF AGARD

According to its Charter, the mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

- Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community;
- Providing scientific and technical advice and assistance to the Military Committee in the field of aerospace research and development (with particular regard to its military application);
- Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;
- Improving the co-operation among member nations in aerospace research and development;
- Exchange of scientific and technical information;
- Providing assistance to member nations for the purpose of increasing their scientific and technical potential;
- Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Programme and the Aerospace Applications Studies Programme. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.

Published March 1988

ISBN 92-835-0451-8

Printed by Specialised Printing Services Limited
40 Chigwell Lane, Loughton, Essex IG10 3TZ

FOREWORD

The present AGARDograph was sponsored by the Computational Fluid Dynamics (CFD) Committee of the Fluid Dynamics Panel recognizing the important role that mesh generation plays in Euler or Navier/Stokes finite difference calculations currently of interest. It has been amply demonstrated that the viability of a numerical solution depends directly on the quality of the mesh as measured by its spacing and orthogonality. Of particular interest is the mesh generation for complex configurations, such as advanced fighters or logistic transports, where a multiblock mesh, for example, is necessary.

There exist numerous reports and books on the various methods of mesh generation giving examples of interest. The present AGARDograph therefore will be directed towards presenting detailed case histories of mesh generation over complex configurations to serve as a guide to users. In particular the emphasis will be on the difficulties encountered, and how they were resolved.

Dr J.Steger, Senior Staff Scientist, at the NASA Ames Research Center and Dr J.Thompson, Professor of Aerospace Engineering at the Mississippi State University, served as the principal authors contributing the Background and Concluding Chapters. Both authors have contributed significantly to the mesh generation research and development. Dr Steger was responsible for the case histories from North America, while Professor Thompson coordinated the contributions from Europe.

The CFD Committee and the Editor wish to express their appreciation to Dr Steger, Professor Thompson and the contributors and their organizations, who generously shared their valuable experiences.

H.Yoshihara
Editor

* * *

AVANT-PROPOS

Le présent AGARDograph a été patronné par le Comite "Calculs de Dynamique des Fluides" (CDF) du Groupe "Dynamique des Fluides", en reconnaissant le rôle important que la génération de mailles joue dans les calculs des différences finies d'Euler ou Navier/Stokes qui présentent actuellement un grand intérêt. Il a été amplement démontre que la viabilité d'une solution numérique dépend directement de la qualité de la maille mesurée par son espacement et son orthogonalité. La génération de mailles est d'un intérêt tout particulier pour des configurations complexes telles que les avions de chasse modernes ou les avions de transport logistiques, dans lesquelles une maille multibloc, par exemple, est nécessaire.

Il existe de nombreux rapports et de nombreux livres sur les différents modes de génération de mailles qui donnent des exemples intéressants. C'est pourquoi le présent AGARDograph aura pour objet de présenter des études de cas de génération de mailles sur des configurations complexes destinées à servir de guide aux utilisateurs. L'accent sera mis en particulier sur les difficultés rencontrées et sur la façon dont elles ont été résolues.

Le Dr J.Steger, Maître de Recherches au Centre de Recherche de la NASA-Ames, et le Dr J.Thompson, Professeur de Techniques Aérospatiales à l'Université de l'Etat du Mississippi sont les principaux auteurs qui ont rédigé les chapitres "Données de base" et "Conclusion". Ces deux auteurs ont contribué considérablement à la recherche et au développement de la génération de mailles. Le Dr Steger était responsable des études de cas provenant d'Amérique du Nord, tandis que le Professeur Thompson assurait la coordination des contributions européennes.

Le Comité chargé de la DFC et le Rédacteur en Chef tiennent à exprimer leurs remerciements au Dr Steger, au Professeur Thompson, ainsi qu'aux autres collaborateurs et à leurs organismes qui ont généreusement apporté une part de leur précieuse expérience.

H.Yoshihara
Rédacteur en Chef

CONTENTS

1. INTENT

Over the last two decades efficient difference schemes for solving the nonlinear governing equations of aerodynamics have evolved for simulating the flow about relatively simple configurations. For the most part these procedures use structured body-conforming curvilinear grids, and are generally being extended to the treatment of more complex shapes by the use of a composite grid approach. However, the development of suitable grid schemes is still an ongoing process, and it is not clear that routine computational fluid dynamics (CFD) solution procedures have evolved, especially for high Reynolds number viscous flow simulation.

For this reason, this AGARDograph was initiated as an attempt to survey some of the capabilities of the CFD community for griding complex three-dimensional configurations. The intent of this AGARDograph is to provide some insight as to the present state of grid generation for aircraft configurations in order to help assess whether this task presents a long term stumbling block to routine use of CFD in aerodynamic applications. At the heart of this AGARDograph are solicited individual contributions describing experience in griding complex configurations for flow simulation.

2. INTRODUCTION

Fluid mechanics is described by nonlinear equations which cannot generally be solved analytically, but which have been solved using various approximate methods including expansion and perturbation methods, sundry particle and vortex tracing methods, collocation and integral methods, and finite difference, finite volume and finite element methods. Generally the finite difference, finite volume and finite element discretization methods have been the most successful, but to use them it is necessary to discretize the field using a grid or mesh. The mesh can be structured or unstructured, but it must be generated under some of the various constraints described below, which can often be difficult to satisfy completely.

The generated mesh must be sufficiently dense that the numerical approximation is an accurate one, but it cannot be so dense that the solution is impractical to obtain. Generally the grid spacing should be smoothly and sufficiently refined to resolve changes in the gradients of the solution. If the grid is also body conforming and curvilinear, the application of boundary conditions is usually simplified. Body-conforming curvilinear grids may also allow the use of various approximate equations such as the boundary-layer equations. The grid should also be constructed with computational efficiency in mind. Various solution algorithms are often highly degraded on grids that are too skewed, too high in aspect ratio, or poorly organized. The accuracy of a numerical approximation can also be impaired if a grid changes discontinuously or is too skewed. Various vectorized computers often require well organized data, and memory requirements can grow to impractical limits unless the data is organized. Finally, the choice of a grid should not lead to overly complex computer codes.

The task of grid generation is not straightforward, given the algorithm and computational constraints imposed by current computers. It is necessary to adapt the grid to the problem at hand to achieve the best efficiency and accuracy. As a result the problem of grid generation can still be as much an art form as it is a scientific discipline.

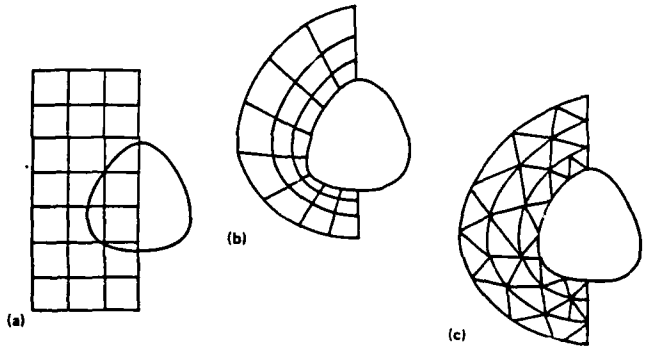
This AGARDograph begins with a brief review of some of the techniques that are available for generating body-conforming curvilinear grids. In order to assess capabilities in grid generation, colleagues at selected institutions were solicited to describe their experiences and difficulties in grid generation of complex configurations. The intent here was not to describe the very latest in grid generation procedures, but to solicit honest comments about what are the difficulties in generating practical grids and what steps are taken to meet these difficulties. These experiences, which comprise the heart of this AGARDograph, are presented as case histories in Section 4.

3. REVIEW

A cursory review of some of the techniques of numerical grid generation is presented below. More information on numerical grid generation and its application to the numerical solution of partial differential equations is given in a recent text on the subject (Ref. 1). Several surveys of the field have also been given (Refs. 2-5), and four conference proceedings dedicated to the area have appeared (Refs. 6-9). The first of these proceedings also contains a number of expository papers and other sources on the subject.

3.1 Grid Types

In the figure below are shown three basic grid treatments for meshing a simple body--a rectangular or Cartesian-like grid, a structured curvilinear body-conforming grid, and an unstructured triangularized grid.



Each grid type has advantages and disadvantages. The rectangular grid is well-ordered, trivial to generate, readily allows accurate interior difference approximations, and the representation of a difference approximation requires the minimum work per step. However, boundary representation requires special logic, is generally of poor accuracy, and the grid does not cluster to efficiently resolve viscous boundary layers on curved boundaries. The curvilinear body-conforming mesh is also well-ordered, allows higher order difference approximations, permits simple and accurate boundary difference approximations, and can be clustered into gradient regions. It is especially well suited for viscous boundary layer approximation. However, the governing equations are more complex to difference on a curvilinear grid (although body-conforming grids often permit use of additional approximations), and grid generation, while not difficult for simple bodies, is no longer trivial. The unstructured triangularized mesh has good grid concentration (i.e., triangles can be readily deleted in smooth gradient regions) and the shape of the boundary curve is readily conformed to. However, such a mesh is poorly ordered and is therefore less amenable to the use of certain algorithms (e.g. ADI) and vectorized computers. Mesh generation is also not trivial. Moreover, triangular meshes have not been used for resolving high Reynolds number viscous boundary layers of practical interest.

For a simple body shape, the use of a single body-conforming curvilinear mesh leads to the most efficient solution procedure. As a result most current aerodynamic solution codes employ a body-conforming structured, curvilinear grid. Considerable effort is now underway to extend these procedures for complex three-dimensional configurations, generally by using composite grid techniques.

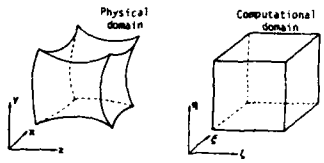
3.2 Grid Structures

A curvilinear structured grid can be represented by a rectangular array of position vectors:

$$\underline{r}_{ijk} \quad (i=1,2,\cdots,I; \quad j=1,2,\cdots,J; \quad k=1,2,\cdots K),$$

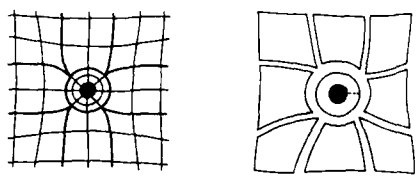
where the indices i,j,k are identified with the three curvilinear coordinates. The position vector $\underline{r}$ is a three-vector giving the values of the x,y,z Cartesian coordinates of a grid point. Since all increments in the curvilinear coordinates cancel out of the transformation relations for derivative operators, there is no loss of generality in defining the discretization to be on integer values of these coordinates.

Fundamental to a body-conforming curvilinear coordinate system is the coincidence of some coordinate surface with each segment of boundary of the physical region. This is accomplished by placing a two-dimensional array of points on a physical boundary segment and setting these values in the array of position vectors with one index constant, e.g. in $\underline{r}_{ijk}$ with i from 1 to I and j from 1 to J. The curvilinear coordinate k is thus constant on this physical boundary segment. With values set on the sides of the rectangular array of position vectors in this manner, the generation of the grid is accomplished by determining the values of $\underline{r}_{ijk}$ in the interior of the rectangular array from the specified boundary values on its sides, e.g. by interpolation or a partial differential equation (PDE) solution. The set of values $\underline{r}_{ijk}$ then forms the nodes of a curvilinear coordinate system filling the physical region. A physical region bounded by six generally curved sides can thus be considered to have been transformed to a rectangular computational region on which the curvilinear coordinates are the independent variables.



3.3 Composite Block Grids

Although in principle it is possible to establish a correspondence between any physical region and a single empty rectangular block for general three-dimensional configurations, the resulting grid is likely to be too skewed and irregular to be usable when the boundary geometry is complicated. A better approach with complicated physical boundaries is to segment the physical region into subregions bounded by six curved sides (four in 2D). These subregions may or may not overlap (c.f. Ref. 10). Each sub-grid is transformed to a rectangular block in the computational region with its own curvilinear coordinate system irrespective of that in the adjacent sub-regions.

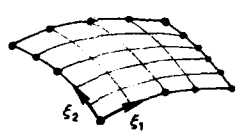


This then allows both the grid generation and numerical solutions on the grid to operate in a rectangular computational region regardless of the shape or complexity of the full physical region. The full region is treated by performing the solution operation in all of the rectangular computational blocks. With the composite framework, CFD solution procedures written to operate on rectangular regions can be incorporated into a code for general configurations in a straightforward manner, since the code only needs to treat a rectangular block. The entire physical field then can be treated in a loop over all the blocks. Such a composite structure has been incorporated in several recent grid codes (e.g. Refs. 11-20 and the papers included in Section 4) of various degrees of generality (cf. also Refs. 9 and 6).

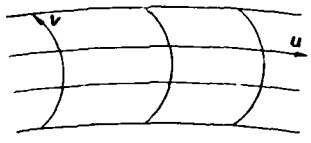
The curved surfaces bounding the sub-regions in the physical region form internal interfaces across which information must be transferred, i.e., from the sides of one rectangular computational block to those of another. Regardless of whether the composite grid is formed using contiguous sub-grids (i.e. a blocked grid) or from overset (or overlapped) grids, these interface boundaries occur in pairs. For a blocked grid an interface on one block is paired with another on the same, or different, block, since both correspond to the same physical surface. Grid lines at the interfaces may meet with complete continuity, with or without slope continuity, or may not meet at all. The codes of Refs. 12, 14, 15, 17, 18, and 19 provide complete continuity, while those of Refs. 16 and 20 are based on slope continuity.

3.4 Surface Grids

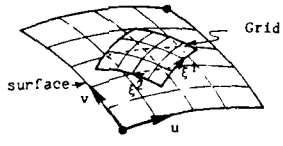
The specification of the boundary point distribution is a two-dimensional grid problem in its own right, which can also be done either by interpolation or a PDE solution. In general, this is a 2D boundary value problem on a curved surface, i.e., the determination of the locations of points on the surface from specified distributions of points on the four edges of the surface.



This is best approached through the use of surface parametric coordinates, whereby the surface is first defined by a 2D array of points, r_{mn}, e.g. a set of cross-sections.



The surface is then splined, and the spline coordinates (u,v; surface parametric coordinates) are then made the dependent variables for the interpolation or PDE generation system. The generation of the surface grid can then be accomplished by first specifying the boundary points in the array r_{ij} on the four edges of the surface grid; converting these Cartesian coordinate values to spline coordinate values (u_{ij},v_{ij}) on the edges; then determining the interior values in the arrays u_{ij} and v_{ij} from the edge values by interpolation or PDE solution; and finally converting these spline values to Cartesian coordinates r_{ij}.



The specification of the 1D point distributions on the edges can be done using certain distribution functions based on hyperbolic functions which have been shown to give spacing distributions that are optimum in the sense of controlling the truncation error induced by spacing changes (cf. Refs. 1,21,22).

After the points on the physical boundary segments have been set on the sides of the rectangular array, the grid is generated throughout the physical region by determinig the interior values in the arrays from the values set on the sides. This amounts to a boundary value problem which can be approached either through interpolation from the boundary values, or through the numerical solution of a system of partial differential equations with r as the dependent variable and the set boundary values as boundary conditions.

3.5 Orthogonality

Coordinate systems that are orthogonal, or at least nearly orthogonal, near the boundary make the application of boundary conditions more straightforward. Although strict orthogonality is not necessary, the accuracy deteriorates if the departure from orthogonality is too large. The implementation of algebraic turbulence models is more reliable with near-orthogonality at the boundary, since information on local boundary normals is usually required in such models. The formulation of boundary-layer equations is also more straightforward and unambiguous in such systems. It is thus better in general, other considerations being equal, for grid lines to be nearly normal to boundaries.

3.6 Grid Generation Schemes

The generation procedures for curvilinear grids are of two general types: (1) by numerical solution of partial differential equations, and (2) construction by algebraic interpolation. In the former, the PDE system may be elliptic, parabolic or hyperbolic. Included in the elliptic systems are both the conformal and the quasiconformal mappings, the former being orthogonal. Orthogonal systems do not have to be conformal, and may be generated from hyperbolic systems as well as from elliptic systems. Some procedures are designed to produce coordinates that are nearly orthogonal. The algebraic procedures include simple normalization of boundary curves, transfinite interpolation from boundary surfaces, the use of intermediate interpolating surfaces, and various other related interpolation techniques.

The relative merits of the various types of grids and generation procedures have been discussed in the various surveys noted above, as well as in the works cited therein. Basically, the algebraic generation systems are faster, but the grids generated from partial differential equations are generally smoother. The hyperbolic and parabolic generation systems are faster than the elliptic systems, but are more limited in the configurations that can be treated. The elliptic systems are the most generally applicable with complicated boundary configurations, but transfinite interpolation is also effective in the composite grid framework.

3.7 Algebraic Grid Generation

Algebraic grid generation consists of the determination of the interior values in the rectangular array $r_{i,j,k}$ from the set values on the sides by interpolation. A number of different forms of interpolation are discussed in Ref. 1. Such generation systems are surveyed in Refs. 2,3 and 5 as well as in Refs. 23 and 24. Here only one widely used procedure, transfinite interpolation, will be briefly described.

A generally effective grid generation procedure is provided by the transfinite interpolation technique (Refs. 25,26), in which all of the boundary values are matched by the interpolation function. Transfinite interpolation in multiple dimensions can be built up of one-dimensional interpolations as follows (cf. Ref. 1 for more details).

One-dimensional interpolation between two boundaries on which the index i is constant is given by

$$r_{i,j,k} = f_i r_{I,j,k} + (1 - f_i) r_{1,j,k} \tag{1}$$

where f_i varies monotonically from $f_1=0$ to $f_I=1$ for $i=1,2,\ldots I$. Analogous forms apply for interpolation in the j and k directions. Certain distribution functions based on hyperbolic functions have been shown to be optimal in the sense of reduced truncation error (cf. Refs. 21,22).

If the interpolation operation given by Eq. (1) is defined as the "projector" $P^{(i)}$, i.e.,

$$r_{ijk} = P^{(i)} = f_i r_{Ijk} + (1 - f_i) r_{1jk} \tag{2}$$

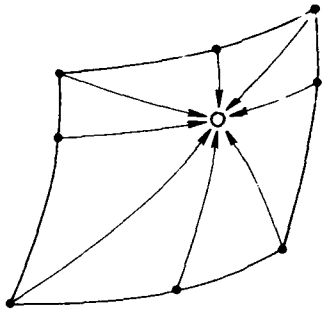
then two-dimensional transfinite interpolation on a surface on which k is constant is accomplished by the projector

$$r_{ijk} = P^{(i)} + P^{(j)} - P^{(i)}P^{(j)} \tag{3}$$

where

$$P^{(i)}P^{(j)} = f_i g_j r_{IJk} + f_i(1 - g_j) r_{I1k} + (1 - f_i) g_j r_{1Jk} + (1 - f_i)(1 - g_j) r_{11k} \tag{4}$$

and g_j varies monotonically from $g_1=0$ to $g_J=1$ for $j=1,2,\cdots,J$.



Analogous forms apply on surfaces on which i or j are constant.

The three-dimensional form then is given by the projector

$$r_{ijk} = P^{(i)} + P^{(j)} + P^{(k)} - P^{(i)}P^{(j)} - P^{(j)}P^{(k)} - P^{(k)}P^{(i)} + P^{(i)}P^{(j)}P^{(k)} \tag{5}$$

where

$$
\begin{aligned}
P^{(I)}P^{(J)}P^{(K)} &= f_i g_j h_k r_{IJK} + f_i g_j (1 - h_k) r_{IJ1} \\
&+ f_i (1 - g_j) h_k r_{I1K} + f_i (1 - g_j)(1 - h_k) r_{I11} \\
&+ (1 - f_i) g_j h_k r_{1JK} + (1 - f_i) g_j (1 - h_k) r_{1J1} \\
&+ (1 - f_i)(1 - g_j) h_k r_{11K} \\
&+ (1 - f_i)(1 - g_j)(1 - h_k) r_{111}
\end{aligned} \tag{6}
$$

Here h_k varies monotonically from $h_1=0$ to $h_K=1$ for k=1,2,--,K.

General algebraic grid generation codes have been reported in Refs. 13, 17, and 27.

3.8 Elliptic Grid Generation

Since elliptic partial differential systems determine a function in terms of its values on the entire closed boundary of a region, such a system can be used to generate the interior values in the array r_{ijk} from the values set on the sides. The properties of elliptic grid generation systems are discussed in Ref. 1. The extremum principles that are exhibited by some elliptic systems serve to prevent the grid overlap that can occur with algebraic grid generation in some configurations. Grids generated from elliptic systems also generally tend to be smoother than those from algebraic systems. In fact, it can be shown by the calculus of variations that a grid generated as the solution of Laplace equations is the smoothest possible grid. The lines of such a grid tend to concentrate over convex portions of the physical boundary and to be more widely spaced over concave portions, however.

Control over the spacing of the grid lines can be exercised by incorporating non-zero Laplacians into the generation system. The most common form at present is the following system:

$$
\sum_{m=1}^{3} \sum_{n=1}^{3} g^{mn} r_{\xi^m \xi^n} + \sum_{n=1}^{3} g^{nn} P_n r_{\xi^n} = 0 \tag{7}
$$

where the g^{mn} are the elements of the contravariant metric tensor:

$$
g^{mn} = \nabla\xi^m \cdot \nabla\xi^n \tag{8}
$$

and the P_n are the "control functions" which serve to control the spacing and orientation of the grid lines in the field. The g^{mn} elements are more conveniently expressed in terms of the elements of the covariant metric tensor, g_{mn}:

$$
g_{mn} = r_{\xi^m} \cdot r_{\xi^n} \tag{9}
$$

which can be calculated directly. Thus

$$
g^{mi} = \frac{1}{g} (g_{ik} g_{j\ell} - g_{i\ell} g_{jk}) \tag{10}
$$

$$
(m,i,j) \text{ cyclic}, \quad (n,k,\ell) \text{ cyclic}
$$

where g, the square of the Jacobian, is given by

$$
g = \det |g_{ij}| = r_{\xi^1} \cdot (r_{\xi^2} \times r_{\xi^3}) \tag{11}
$$

In these relations, r is the Cartesian position vector of a grid point ($r = ix + jy + kz$), and the ξ^i (i=1,2,3) are the three curvilinear coordinates.

Negative values of the control function P_n cause grid lines on which ξ^n is constant to tend to move in the direction of decreasing ξ^n, and this feature can be used to concentrate grid lines near other grid lines and/or points or in certain regions in physical space. However, a more automatic procedure is to determine the control functions so as to reflect the boundary point spacing into the field. (Laplace equations, i.e., with zero control functions, tend to produce uniform grids in the field regardless of the concentration of points on the boundary.) This is accomplished as follows (cf. Ref. 1 for details of the development).

In two dimensions, the projection of Eq. (7) along a coordinate line on which i (i.e. ξ^1) varies yields the following equation for the control function P_1 on this line:

$$P_1 = -S_1 + \frac{|\underline{r}_{\xi^1}|}{\rho_1} \tag{12}$$

The first term here,

$$S_1 = \frac{\underline{r}_{\xi^1} \cdot \underline{r}_{\xi^1\xi^1}}{|\underline{r}_{\xi^1}|^2} \tag{13}$$

contains only derivatives along the line, and hence can be evaluated from the point distribution on the line. This term is the logarithmic derivative of arc length along the line. In the last term ρ_1 is the radius of curvature of the line on which ξ^1 is constant that crosses the line on which the control function P_1 is to be evaluated. This curvature is given by

$$\rho_1 = -\left[\frac{\underline{n}_2 \cdot \underline{r}_{\xi^2\xi^2}}{|\underline{r}_{\xi^2}|^2}\right]^{-1} \tag{14}$$

where $\underline{n}_2$ is the unit normal to the crossing line. Although $|\underline{r}_{\xi^1}|$, the arc length spacing along the line of evaluation, can be evaluated from the point distribution on that line, the radius of curvature requires derivatives off that line. Analogous equations apply for the evaluation of the control function P_2 on a line on which j, i.e., ξ^2, varies.

The arc length contribution, S_1, and the arc spacing, $|\underline{r}_{\xi^1}|$, of the control function P_1 are evaluated on the two edges (j=1 and j=J) on which i varies. The radius of curvature, ρ_2 is also evaluated on these lines. These evaluations use

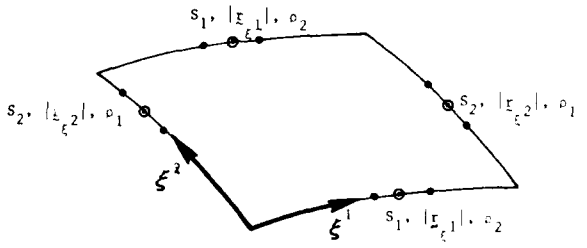
$$\underline{r}_{\xi^1}^{(J)} = \frac{1}{2}\left(\underline{r}_{i+1,J} - \underline{r}_{i-1,J}\right)$$

$$\underline{r}_{\xi^1\xi^1}^{(J)} = \underline{r}_{i+1,J} - 2\underline{r}_{iJ} + \underline{r}_{i-1,J}$$

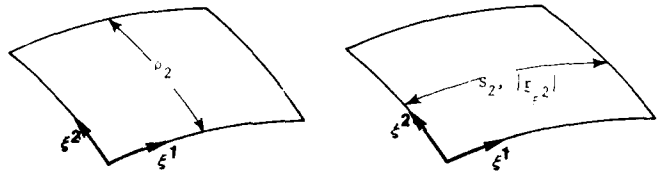
on the j=J line, with analogous expressions on the j=1 line. The normal, $\underline{n}_1$, needed for the evaluation of ρ_2 is

$$\underline{n}_1 = \underline{k} \times \frac{\underline{r}_{\xi^1}}{|\underline{r}_{\xi^1}|}$$

where $\underline{k}$ is the unit vector normal to the surface. Similarly, the arc length contribution to P_2, the radius of curvature ρ_1 and the spacing $|\underline{r}_{\xi^2}|$ are evaluated on the other two edges (i=1 and i=I).



The control functions in the interior are then evaluated by interpolating the components S_1, $|r_{\xi_1}|$, and ρ_2 one-dimensionally in the j-direction from the two edges on which they have been evaluated, i.e., the j=1 and j=J lines. Similarly S_2, $|r_{\xi_2}|$, and ρ_1 are interpolated in the i-direction between the i=1 and i=I lines.



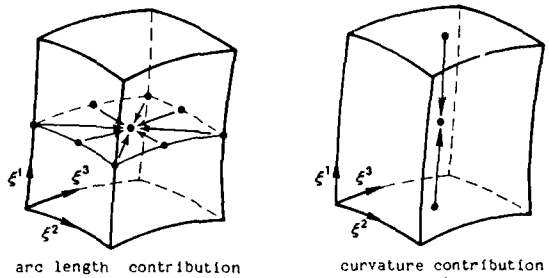
The control functions then are formed from these interpolated values:

$$P_1 = -S_1 + \frac{|r_{\xi_1}|}{\rho_1} \tag{15}$$

$$P_2 = -S_2 + \frac{|r_{\xi_2}|}{\rho_2} \tag{16}$$

In three dimensions the arc length contribution, S_1, and the arc spacing, $|r_{\xi_1}|$, are evaluated on the four sides of the computational block on which i varies, i.e., the sides j=1 and j=J and the sides k=1 and k=K. The radius of curvature, ρ_1, is evaluated on the two sides on which i is constant (i=1 and i=I) from the relation

$$\rho_1 = -\left[\frac{\underline{n}_2 \cdot \underline{r}_{\xi_2\xi_2}}{|r_{\xi_2}|^2} + \frac{\underline{n}_3 \cdot \underline{r}_{\xi_3\xi_3}}{|r_{\xi_3}|^2}\right]^{-1} \tag{17}$$

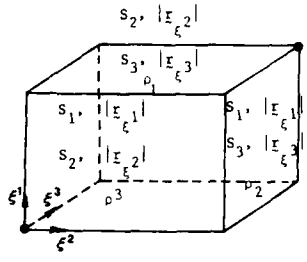


arc length contribution for P_1

curvature contribution for P_1

Analogous evaluations are done for S_2 and $|r_{\xi_2}|$ on the four sides on which j varies, for S_3 and $|r_{\xi_3}|$ on the four sides on which k varies, for ρ_2 on the two sides on which j is constant, for ρ_3 on the two on which k is constant. Then S_1 and $|r_{\xi_1}|$ are interpolated two-dimensionally in the j and k directions from the four sides on which i varies using transfinite inter-

polation, and ρ_1 is interpolated one-dimensionally in the i direction from the two sides on which i is constant.



The control function P_1 is then evaluated from

$$P_1 = -S_1 + \frac{|r_{\xi 1}|}{\rho_1} \tag{18}$$

Analogous interpolations allow the evaluation of the other two functions.

General codes based on such elliptic generation systems appear in Refs. 12, 14-20. The code of Refs. 16,20 uses an iterative adjustment of control functions to achieve boundary orthogonality, as can the code of Ref. 17, as follows:

A second-order elliptic generation system allows either the point locations on the boundary or the coordinate line slope at the boundary to be specified, but not both. It is possible, however, to iteratively adjust the control functions in the generation system of the Poisson type discussed above until, not only a specified line slope, but also the spacing of the first coordinate surface off the boundary is achieved, with the point locations on the boundary specified (cf. Ref. 16).

In three dimensions the specification of the coordinate line slope at the boundary requires the specification of two quantities, e.g., the direction cosines of the line with two tangents to the boundary. The specification of the spacing of the first coordinate surface off the boundary requires one more quantity, and therefore the three control functions in the system Eq. (7) are exactly sufficient to allow these three specified quantities to be achieved, while the one boundary condition allowed by the second-order system provides for the point locations on the boundary to be specified.

To illustrate this development, an iterative procedure can be constructed for the determination of the control functions in two dimensions as follows (cf. Ref. 16). Consider the generation system given by Eq. (7) in two dimensions (with $\xi^1 = \xi$, $\xi^2 = \eta$, $x_1 = x$, and $x_2 = y$ here). On a boundary segment that is a line of constant η, r_ξ and $r_{\xi\xi}$ are known from the specified boundary point distribution. Also $|r_\eta|$, the spacing off this boundary, is specified, as is the condition of orthogonality at the boundary, i.e., $r_\xi \cdot r_\eta = 0$. But specification of $|r_\eta| = \sqrt{x_\eta^2 + y_\eta^2}$, together with the condition $r_\xi \cdot r_\eta = x_\xi x_\eta + y_\xi y_\eta = 0$ provides two equations for the determination of x_η and y_η in terms of the already known values of the x_ξ and y_ξ. Therefore r_η is known on the boundary.

Because of the orthogonality at the boundary, Eq. (7) reduces to the following equation on the boundary:

$$|r_\eta|^2(r_{\eta\eta} + Pr_\xi) + |r_{\xi\xi}|^2(r_{\xi\xi} + Qr_\eta) = 0 \tag{19}$$

Dotting r_ξ and r_η into this equation, and again using the condition of orthogonality, yields the following two equations for the control functions on the boundary:

$$P = -\frac{\underline{r}_\xi \cdot \underline{r}_{\xi\xi}}{|\underline{r}_\xi|^2} - \frac{\underline{r}_\xi \cdot \underline{r}_{\eta\eta}}{|\underline{r}_\eta|^2} \tag{20}$$

$$Q = -\frac{\underline{r}_\eta \cdot \underline{r}_{\eta\eta}}{|\underline{r}_\eta|^2} - \frac{\underline{r}_\eta \cdot \underline{r}_{\xi\xi}}{|\underline{r}_\xi|^3} \tag{21}$$

All of the quantities in these equations are known on the boundary except $\underline{r}_{\eta\eta}$. (On a boundary that is a line of constant ξ, the same equations for the control functions result, but now with $\underline{r}_{\xi\xi}$ the unknown quantity.

The iterative solution thus proceeds as follows:

(1) Assume values for the control functions on the boundary.

(2) Solve Eq. (7) to generate the grid in the field.

(3) Evaluate $\underline{r}_{\eta\eta}$ on η-line boundaries, and $\underline{r}_{\xi\xi}$ on ξ-line boundaries, from the result of Step (2), using one-sided difference representations. Then evaluate the control functions on the boundary from Eqs. (20) and (21). Evaluate the control functions in the field by interpolation from the boundary values.

Steps (2) and (3) are then repeated until convergence.

The analogous procedure for three dimensions is given in Refs. 20, 17, and 1.

3.9 Unstructured Meshes

An alternative to the structured quadralateral meshes that are discussed in this report are the unstructured meshes composed of triangles in 2D or tetrahedrons in 3D (Ref. 28). The unstructured mesh requires less ingenuity to devise (though not necessarily to code) for complicated regions than does the structured mesh, but requires considerably more computer time and storage, as well as a much more involved data handling procedure. Combinations of structured and unstructured meshes can also be used, with structured meshes near the boundaries connected by unstructured meshes (Ref. 29).

3.10 Adaptive Grid Schemes

Finally, dynamically-adaptive grids continually adapt to follow developing gradients in the physical solution. This adaption can reduce the oscillations associated with inadequate resolution of large gradients, allowing sharper shocks and better representation of boundary layers. Another advantageous feature is the fact that in the viscous regions where real diffusion effects must not be swamped, the numerical dissipation from upwind biasing is reduced by the adaption. Dynamic adaption is at the frontier of numerical grid generation and may well prove to be one of its most important aspects, along with the treatment of real three-dimensional configurations through the composite grid structure.

There are three basic strategies that may be employed in dynamically adaptive grids (cf. Refs. 1,4) coupled with the partial differential equations of the physical problem. (Combinations are also possible, of course.):

(1) Redistribution of a fixed number of points.

In this approach, points are moved from regions of relatively small error or solution gradient to regions of large error or gradient. As long as the redistribution of points does not seriously deplete the number of points in other regions of possible significant gradients, this is a viable approach. The increase in spacing that must occur somewhere is not of practical consequence if it occurs in regions of small error or gradient, even though in a formal mathematical sense the global approximation is not improved. The redistribution approach has the advantage of not increasing the computer time and storage during the solution, and of being straightforward in coding and data structure. The disadvantages are the possible deleterious depletion of points in certain regions and the possibility of the grid becoming too skewed.

Recent examples of this adaptive approach in CFD are Ref. 30 in 2D and Ref. 31 in 3D.

(2) Local refinement of a fixed set of points.

In this approach, points are added (or removed) locally in a fixed point structure in regions of relatively large error or solution gradient. Here there is, of course, no depletion of points in other regions and therefore no formal increase of error occurs. Since the error is locally reduced in the area of refinement, the global error does formally decrease. The practical advantage of this approach is that the original point structure is preserved. The disadvantages are that the computer time and storage increase with the refinement, and that the coding and data structure are difficult, especially for implicit flow solvers.

Recent examples of this adaptive approach in CFD are Ref. 32 and 33, both in 2D.

(3) Local increase in algorithm order.

In this approach, the solution method is changed locally to a higher-order approximation in regions of relatively large error or solution gradient without changing the point distribution. This again increases the formal global accuracy since a local increase is achieved without an attendant decrease in formal accuracy elsewhere. The advantage is that the point distribution is not changed at all. The disadvantage is the great complexity of implementation in implicit flow solvers.

This adaptive approach has not had any significant application in CFD in multiple dimensions.

Adaptive redistribution of points traces its roots to the principle of equidistribution of error (cf. Ref. 1,4) by which a point distribution is set so as to make the product of the spacing and a weight function constant over the points:

$$w\Delta x = \text{constant} \tag{22}$$

With the point distribution defined by a function $x(\xi)$, where ξ varies by a unit increment between points, the equidistribution principle can be expressed as

$$wx_\xi = \text{constant} \tag{23}$$

This one-dimensional equation can be applied in each direction in an alternating fashion, but a direct extension to multiple dimensions can be made in either of two ways as follows:

From the calculus of variations, Eq. (23) can be shown (cf. Ref. 1) to be the Euler variational equation for the function $x(\xi)$ which minimizes the integral

$$I = \int w(\xi)x_\xi^2 \, d\xi \tag{24}$$

Generalizing this, a competitive enhancement of grid smoothness, orthogonality, and concentration can be accomplished by representing each of these features by integral measures over the grid and minimizing a weighted average of the three. This approach was put forward in Ref. 30 and is discussed in detail in Ref. 1.

The second approach is to note the correspondence between Eq. (23) and the one-dimensional form of the following commonly-used elliptic grid generation system, Eq. (7). Here the "control functions", P_n, serve to control the grid line spacing and orientation. The 1D form of this system is

$$x_{\xi\xi} + Px_\xi = 0 \tag{25}$$

Differentiation of Eq. (23) yields

$$wx_{\xi\xi} + w_\xi x_\xi = 0 \tag{26}$$

Then, from Eq. (25) and (26),

$$\frac{x_{\xi\xi}}{x_\xi} = -P = -\frac{w_\xi}{w} \, . \tag{27}$$

from which the control function can be taken as

$$P = \frac{w_\xi}{w} \tag{28}$$

It is logical then to represent the control functions in 3D as

$$P_n = \frac{w_{\zeta_n}}{w} \qquad n = 1,2,3 \tag{29}$$

This approach was put forward in Ref. 34 and has been applied in 3D in Ref. 31.

3.11 References

1. THOMPSON, J.F., WARSI, Z.U.A., and MASTIN, C.W., Numerical Grid Generation: Foundations and Applications, North-Holland, 1985.

2. THOMPSON, J.F., WARSI, Z.U.A., and MASTIN. C.W., "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review", Journal of Computational Physics, Vol. 47, p. 1, 1982.

3. THOMPSON, J.F., "Grid Generation Techniques in Computational Fluid Dynamics", AIAA Journal, Vol. 22, p. 1505, 1984.

4. THOMPSON, J.F., "A Survey of Dynamically-Adaptive Grids in the Numerical Solution of Partial Differential Equations", Applied Numerical Mathematics, Vol. 1, p. 3, 1985.

5. EISEMAN, P.R., "Grid Generation for Fluid Mechanics Computations", Annual Review of Fluid Mechanics, Vol. 17, 1985.

6. THOMPSON, J.F., (Ed.), Numerical Grid Generation, North-Holland 1982. (Also published as Vol. 10 and 11 of Applied Mathematics and Computation, 1982).

7. SMITH, ROBERT E., (Ed.), Numerical Grid Generation Techniques, NASA Conference Publication 2166, NASA Langley Research Center, 1980.

8. GHIA, K.N., and GHIA, U., (Ed.), Advances in Grid Generation, FED-Vol. 5, ASME Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Houston, 1983.

9. HAUSER, J. and TAYLOR, C., (Ed.) Numerical Grid Generation in Computational Fluid Dynamics, Pineridge Press, 1986.

10. THOMPSON, J.F., (Ed.), "A Survey of Composite Grid Generation for General Three-Dimensional Regions," in Numerical Methods for Engine-Airframe Integration, S.N.B. Murthy and G.C. Paynter, (Ed.), AIAA, 1986.

11. THOMAS, P.D., "Composite Three-Dimensional Grids Generated by Elliptic Systems", AIAA Journal, Vol. 20, p. 1195, 1982.

12. COLEMAN, R.M., "INMESH: An Interactive Program for Numerical Grid Generation", DTN-SRDC-85-054, David W. Taylor Naval Ship Research and Development Center, Bethesda, MD, 1985.

13. SONI, B.K., "Two- and Three-Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics", AIAA-85-1526, AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, Ohio, 1985.

14. MIKI, KAZUYOSHI and TAKAGI, TOSHIYUKI, "A Domain Decomposition and Overlapping Method for the Generation of Three-Dimensional Boundary-Fitted Coordinate Systems", Journal of Computational Physics, Vol. 53, p. 319, 1984.

15. WEATHERILL, N.P. and FORSEY, C.R., "Grid Generation and Flow Calculations for Complex Aircraft Geometries Using a Multi-Block Scheme", AIAA-84-1665, AIAA 17th Fluid Dynamics, Plasma Dynamics, and Laser Conference, Snowmass, Colorado, 1984.

16. SORENSON, R.L., and STEGER, J.L., "Grid Generation in Three Dimensions by Poisson Equations with Control of Cell Size and Skewness at Boundary Surfaces", Advances in Grid Generation, FED-Vol. 5, (Ed.) K.N. Ghia and U. Ghia, ASME Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Houston, 1983.

17. THOMPSON, J.F., "A Composite Grid Generation Code for General 3-D Regions", AIAA-87-0275, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

18. WOAN, C.J., "Three-Dimensional Elliptic Grid Generations Using a Multi-Block Method", AAA-87-0278, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

19. HOLCOMB, J.E., "Development of a Grid Generator to Support 3-D Multizone Navier-Stokes Analysis", AIAA-87-0203, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

20. SORENSON, R.L., "Three-Dimensional Elliptic Grid Generation About Fighter Aircraft for Zonal Finite-Difference Computations", AIAA-86-0429, AIAA 24th Aerospace Sciences Meeting, Reno, 1986.

21. VINOKUR, MARCEL, "On One-Dimensional Stretching Functions for Finite-Difference Calculations", Journal of Computational Physics, 50, 215, 1983.

22. THOMPSON, J.F., and MASTIN, C.W., "Order of Difference Expressions in Curvilinear Coordinate Systems", J. Fluids Eng., 107, 241, 1985.

23. SMITH, R.E., "Algebraic Grid Generation", Numerical Grid Generation, (Ed.) Joe F. Thompson, North-Holland, p. 137, 1982.

24. SMITH, R.E., "Three-Dimensional Algebraic Grid Generation," AIAA-83-1904, AIAA Computational Fluid Dynamics Conference, Danvers, Mass., 1983.

25. GORDON, WILLIAM J. and THIEL, LINDA C., "Transfinite Mappings and Their Application to Grid Generation", Numerical Grid Generation, (Ed.) Joe F. Thompson, North-Holland, p. 171, 1982.

26. RIZZI, A. and ERIKSSON, L.E., "Transfinite Mesh Generation and Damped Euler Equation Algorithm for Transonic Flow around Wing-Body Configurations", AIAA-81-0999, AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, California, 1981.

27. EISEMAN, PETER R., "Automatic Algebraic Coordinate Generation", Numerical Grid Generation, (Ed.) Joe F. Thompson, North-Holland, p. 447, 1982.

28. JAMESON, A. and BAKER, T.J., "Improvements to the Aircraft Euler Method", AIAA-87-0452, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

29. NAKAHASHI, K. and OBAYASHI, S., "FDM-FEM Zonal Approach for Viscous Flow Computations Over Multiple-Bodies", AIAA-87-0604, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

30. BRACKBILL, J.U. and SALTZMAN, J.S., "Adaptive Zoning for Singular Problems in Two Dimensions", Journal of Computational Physics, 46, 342, 1982.

31. KIM, H.J., and THOMPSON, J.F., "Three Dimensional Adaptive Grid Generation on a Composite Block Grid", to be presented at the AIAA 26th Aerospace Sciences Meeting, Reno, January 11-14, 1988.

32. DANNENHOFFER, J.F. III, and BARON, J.R., "Grid Adaptation for the 2-D Euler Equations", AIAA-85-0484, AIAA 23rd Aerospace Sciences Meeting, Reno, 1985.

33. *ODEN, J.T., STROUBOULIS, T., and DEVLOO, P., "An Adaptive Finite Element Strategy for Complex Flow Problems", AIAA-87-0557, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.*

34. ANDERSON, D.A., "Generating Adaptive Grids with a Conventional Grid Scheme", AIAA-86-0427, AIAA 24th Aerospace Sciences Meeting, Reno, 1986.

4. CONTRIBUTIONS

4.1 Solicitation and Overview

The essence of this AGARDograph is a compilation of case histories depicting the current state of the art in grid generation activities for complex configurations. Researchers who have been actively building and using grid generation codes for simulating 3D configurations were solicited to describe their experiences in this area. In addition to the usual method description, each contributor was asked to describe what grid topologies they have used and what steps they took to generate a grid for a particular application. The contributors were also asked to provide some details of what were the most time consuming tasks, what difficulties they encountered and how these were resolved, and what steps they might take in the future to improve matters or what software is needed to better generate grids.

None of the papers contributed to this AGARDograph used unstructured grids or rectangular grids with unstructured boundary/interface elements. Nevertheless, they give a fairly accurate picture of the capability for griding complex configurations as of about the first half of 1987. The overall impression drawn from these contributions is that considerable capability exists to adequately mesh relatively complex shapes given adequate time (measured in weeks). This technology, or an alternative, must continue to improve, however, or it will indeed impede the long term goal of complete aircraft simulation.

The papers contributed to this AGARDograph convey a wealth of information but here we wish to note only two points. The first point, mentioned in the introduction, is that most of the contributed papers have adopted the composite grid approach to griding complex configurations, thereby allowing an extension of existing single-grid curvilinear grid algorithms. Moreover, the most prevalent approach is the blocked (i.e., non-overlapped) grid method. Nine of the eleven contributed papers deal with generating a grid for general purpose simulation of flow about complex configurations (the papers contributed by Yoshihara and by Sobieczky having more limited objectives). Of these, only the approaches described by Eberle and by Benek did not explicitly adopt the blocked grid approach, although the Benek approach can include blocked grid boundaries.

Among the advantages cited for the composite grid approach are the following:

(1) ease of treatment of complex configurations.
(2) capability for local refinement and modification.
(3) reduced core storage.
(4) natural use of different flow equations in different regions.

A second point is that because of the emphasis on composite grids, the tasks of subdividing the grids, generating surface grids, and providing interfaces have become more time consuming and critical than the task of generating the interior grids. The contributed papers on composite grids either strongly hint at, or explicitly note, that how a grid should be subdivided depends on the geometry, the numerical algorithm used, the flow features, etc. So, given a limited computer resource, the sub-grids of a composite grid must be selected with care. This implies a learning process and a need for human interaction. Like geometry definition, the tasks of subgriding, interfacing, and surface grid definition are being assigned to interactive workstations. Various levels of sophistication in treating these problems in this way are evident in the contributed papers. What is strongly implied is that these are not simple tasks or ones for which off-the-shelf software is available. This is evidently a pacing area of research in complex grid generation.

Surface grid generation is seen to have a dominant effect on the quality of the volume grid, to be very time-consuming, and to be in considerable need of improvement in regard to the specification of boundary data sets and the interactive manipulation thereof. There is a feeling that more emphasis should be put on the development of CAD geometry tools especially suited to the needs of CFD.

The topological definition of the block structure is seen to require considerable experience and to be difficult to teach. There is a need for automation of this process, perhaps through the use of artificial intelligence or other means.

The critical need for graphical interaction, especially in regard to surface grid generation, block definition, and grid control is evident. The process of grid generation for complex configurations still requires too large an amount of man-time.

It appears now that the theoretical developments necessary for effective grid generation are largely in hand, but that a very large amount of effort is still needed in the efficient implementation of the processes.

4.2 LESSONS LEARNED IN THE MESH GENERATION FOR PN/S CALCULATIONS

H. Yoshihara
Boeing Military Airplane Company
Seattle, WA, USA

SUMMARY

Experiences encountered in the 2D mesh generation with the elliptic differential equation method are described for the PN/S calculations over a generic fighter at a supersonic Mach number and for a wing/fuselage at hypersonic Mach numbers. Importance of the mesh quality is stressed, and the need of an improved cost-effective treatment of the shocks is pointed out.

1. INTRODUCTION

Experience has amply demonstrated that accurate finite difference solutions can only be achieved if the mesh is sufficiently refined and orthogonal. That is, the quality of the mesh impacts both the stability and accuracy of the numerical solutions with the sensitivity depending on the difference equations and boundary conditions on hand and on the solution algorithm. An efficient mesh will finally impact directly the computer costs. In the following we shall recollect the experiences encountered in the mesh generation for the parabolized Navier/Stokes (PN/S) calculations for the supersonic flow over a generic fighter (Ref. 1) and for the hypersonic flow over a generic wing/fuselage (Ref. 2). Features of the solution and solution procedure relevent to the mesh generation will be described.

The PN/S equations are based on the Reynolds-averaged thin-layer N/S equations in which the pressure is assumed to be constant across tne subsonic portion of the boundary layer (the sublayer approximation). Closure is achieved using the Baldwin/Lomax mixing-length turbulence model. When the inviscid portion of the flow is supersonic, the solution can be obtained by a spatial marching in the free stream direction. In the cases to be calculated the nose shock from the fuselage apex is fitted, but shock waves occurring further downstream are captured. In comparison to the the unsteady N/S procedure, with the PN/S procedure, the computer time is greatly reduced; but, more importantly, the mesh generation is greatly simplified, requiring only a two dimensional (2D) mesh generation. The latter then permitted the treatment of the complex fighter configuration. There is however a disadvantage introduced by the sublayer approximation. There arises a numerical stability limitation requiring the streamwise marching step to be "much larger" than the height of the subsonic sublayer.

For the 2D mesh generation, the Steger/Sorenson elliptic differential equation method was used. It entails the solution of two decoupled boundary value problems for the dependent variables $r = r(\eta,\zeta)$ and $\theta = \theta(\eta,\zeta)$. Here r and θ are the polar coordinates in a marching plane in the physical space, whereas η and ζ are the cartesian coordinates in the computational plane in which the physical space in a marching plane is mapped to the interior of a unit square. (See Figure 1.) The two elliptic equations for $r(\eta,\zeta)$ and $\theta(\eta,\zeta)$ contain non-homogeneous terms which are chosen to impose orthogonality of the mesh in the neighborhood of the boundaries. Boundary conditions on the unit square are posed to obtain the desired mesh topology and spacing. The above boundary value problems were solved by a point-relaxation code furnished by Dr. Denny Chaussee of NASA-Ames.

2. THE SUPERSONIC CASE-MODEL 350 GENERIC FIGHTER (REF. 1)

Calculations were carried out on the configuration shown in Figure 2 for a free stream Mach number of 2.2, at 10 degrees angle of attack, and assuming a turbulent flow. The marching coordinate x was taken as the body-oriented axis passing through the fuselage nose. In general 91 mesh points were used on the half-circumference and 45 points in the radial direction. For the radial spacing, the location of the first point off the solid surfaces was chosen; and the spacing of further outboard points was geometrically stretched with a geometric ratio that located the outermost mesh point on the outboard boundary formed by the nose shock. The location of the first point and the total number of "radial" points were selected such that there was a minimum of five points within the subsonic portion of the boundary layer. With the height of the subsonic sublayer varying over a wide range along the configuration cut, the location of the first point off the surface should be correspondingly varied to achieve an efficient mesh. However in the above calculations, the location of the first point was fixed at the level to accommodate the smallest subsonic sublayer height, resulting in excessive refinement elsewhere.

Selection of the Boundary conditions

The selection of the boundary conditions for the mesh generation problem is an important first step which establishes the topology and spacing of the mesh and indirectly affects the orthogonality of the mesh in the domain interior. In general

the procedure to establish the boundary conditions is to locate the boundary mesh points in the physical space to achieve the desired spacing. The resulting values of r and θ are then assigned as the boundary values at the corresponding points in the computational plane. (In the latter the mesh points are distributed uniformly along each coordinate line.) Along the configuration cut (ABCD), the mesh points were distributed to fit the expected flow gradients. Thus for example the points would be clustered about the sharp leading edge (Point C). Along the symmetry boundaries, AG and DE, the mesh points were distributed with the geometric stretching described earlier. Less obvious was the appropriate mesh placement along the shock EFG. Here the mesh points were distributed with the same relative spacing used along the configuration cut ABCD. Some tuning of the above boundary conditions was usually required to obtain the final mesh.

The boundary conditions, as determined above, must be updated as the marching progresses downstream, since the configuration cut can change greatly, in general necessitating both an addition as well as a redistribution of the mesh points. Without adequate control, a clustered mesh about the wing tip drifted from its intended location about the tip to the wing surface, resulting in a severe skewing of the mesh by the misalignment of the end points of the radial mesh lines.

The elliptic boundary value problems with the above boundary conditions were solved by point relaxation, and a typical mesh at a wing station is shown in Figure 3. Here the convergence of the relaxation process must be carried out to a point that the desired mesh orthogonality near the configuration surface has been achieved.

Treatment of the Swept Canard Trailing Edge

The marching plane used was a plane of constant x. The use of this marching plane offered no difficulties until the swept trailing edge was intercepted as at Station X of Figure 4. Here the configuration cut assumed the multiply connected domain shown on the lower left part of Figure 4. The mesh for this configuration can be generated in the usual way if the wake segment BD and GF are placed on the two sides of a slit. The corresponding computational plane assumes the configuration shown on the right part of Figure 4. The difficulty for the numerical formulation is that the flow continuity condition across the slit envolves two points lying on separate line segments. With an implicit treatment of the continuity condition, the simple solution procedure used previously is no longer possible.

The above difficulty can be circumvented in several ways. In the case of a modest trailing edge sweep, a shearing transformation can be made to unsweep the trailing edge. The usual marching procedure can then be used across the trailing edge. The procedure used in Ref. 1 however was to bridge the trailing edge by solving the intervening domain containing the trailing edge by the unsteady N/S code ARC3D. Here the solution at the slit points was determined using a one-sided difference, and the resulting unequal values at the corresponding slit points were simply averaged for use as initial data for the next time step. Upon convergence, the continuity of the solution at the slit would be achieved. The supersonic outflow condition was prescribed at the downstream boundary. The 3D mesh for the ARC3D solution was generated by interpolating a sequence of 2D meshes, one of which shown in Figure 5. The ARC3D solutions on the two most downstream planes were discarded, and the solution on the next two further upstream planes were used as initial data for the further PN/S marching.

Treatment of the Underwing Nacelles

The inlet faces of the 350 configuration lie on a constant-x plane, so that no difficulties were encountered in the march onto the nacelle. The marching was first carried out downstream to the inlet face, yielding the solution on the mesh shown in the upper part of Figure 6 containing the nacelle centerbody. A new mesh was then generated at this station with the configuration cut now containing the nacelle highlight (lower part of Figure 6). The solution on the upstream mesh (upper mesh) was then interpolated onto the downstream mesh (lower mesh) to yield the initial data for the further downstream marching.

Mesh Refinement About the Wing Tip

For the supersonic cases with the leading edge only slightly subsonic, the leading edge pressures, both on the upper and lower surfaces, should be reasonably well behaved without sharp suction or overpressure peaks. In Figure 7 the spanwise pressure distribution at Station D is shown where a sharp peak occurred on the lower surface near the leading edge. This is to be contrasted to the expected smooth distribution obtained at Station B also shown in Figure 7. The cause of the suspect peak at Station D is due to the truncation errors associated with both the flow solution and the numerical determination of the near-singular Jacobian in the neighborhood of the "pointed" leading edge. The sharp spike can only be removed by using a sufficiently refined orthogonal mesh about the leading edge. (Here a "conservative" differencing of the Jacobian offered no relief.)

3. THE HYPERSONIC CASE-GENERIC WING/FUSELAGE (REF. 2)

PN/S calculations were also carried out for the generic wing/fuselage shown in Figure 8 at Mach numbers of 10 and 25 and at zero angle of attack assuming the boundary layer to be turbulent. Fluid dynamically these hypersonic flows are more extreme relative to the previous supersonic cases due to the appearance of greatly strengthened shock waves and increased heating effects. In general the mesh generation procedure followed that used in the above supersonic case, but the requirements on the mesh quality were more stringent due to the increased "severity" of the flow. A typical mesh of dimension 101 (half-circumference) x 60 (radial) generated with the above elliptic method is shown in Figure 9. The first mesh point from the surface over most of the configuration was located at 0.02 inches, but closer locations were used in the nose and wing leading edge regions. The axial marching step was 0.5 inches. (These dimensions are to be viewed relative to the fuselage length of 1235 inches.) In the following, additional facets of the mesh generation process arising by the increase of the Mach number are described.

The Temperature Boundary Layer

In Figure 10 radial profiles of the velocity and temperature at the crest point at Station 533 are shown for a turbulent flow at M = 10. Here a surface temperature of 1800° R was prescribed. Of particular significance is the highly-peaked temperature overshoot arising in the lower part of the boundary layer generated by the intense eddy dissipation. It is clear by examining these profiles that the adequacy of the mesh refinement in the boundary layer is dictated, not by the more customary velocity profile, but by the temperature profile.

Shock Capture

In the present calculations the nose shock from the fuselage apex was fitted, but shocks arising further downstream, as the wing bow shock, were captured. In the shock capture process an appropriately refined mesh must be used to obtain an acceptable shock thickness. In the case of an inclined shock cutting across the mesh, a fine mesh must be used in all coordinate directions cutting across the shock. Thus if the shock is aligned with the "transverse" mesh lines, a fine mesh would be required only in the direction cutting across the shock.

The leading edge radius of the wing was 0.5 inches (this to be contrasted to the wing root chord of the order of 500 inches). With a streamwise marching step of 0.5 inches, the question then arises as to how the detached shock flow about the leading edge with a radius equal to the marching step could be adequately resolved. The answer is that the detached shock flow is essentially "aligned" with the leading edge, and the marching in the computational plane takes place essentially in the direction of the leading edge. That is, the coordinate lines are aligned with the strong portion of the detached shock. The resolution of the detached shock flow is then dictated, not by the 0.5 inch marching step, but by the more highly refined transverse mesh in the marching plane. Away from the leading edge region, this alignment no longer prevails and with the transverse mesh itself coarsened, the shock will assume a much larger thickness.

For the proper design of external compression inlets, which might be incorporated on the wing lower surface of the above configuration, it is essential that the shocks from the compression ramps be captured with a sufficiently small thickness. Also to obtain the proper interaction of the wing shock on the thick fuselage boundary layer, it is again important to capture the wing shock with a sufficiently small thickness. In a 3D problem, one would turn to a 3D adaptive mesh program to align the mesh with the shocks as well as bunch the mesh lines in the direction normal to the shock. The present PN/S procedure is a 2D method, solving the flow only in a cross-flow marching plane. Use of an adaptive mesh program in this marching plane clearly would not achieve our goal completely. The x-marching would still cut across such aligned shocks, necessitating a refined marching step.

Shock-on-Shock Interaction

As one marches sufficiently far downstream, the nose shock from the fuselage apex will approach the wing leading edge and intersect the wing detached shock. A difficulty now arises as the 60 radial mesh points are squeezed into an ever-decreasing interval as the shock (the outer boundary) approaches the wing leading edge (the inner boundary). Moreover, computational difficulty can be anticipated as the fitted nose shock approaches and intersects the captured wing shock, and the treatment of the wing shock is switched from a capture to a fitting procedure as it emerges as the most upstream shock. To circumvent these difficulties, the treatment of the nose shock was switched from a fitting to a capturing procedure just upstream of its intersection with the wing shock. In this switch, the outer boundary was selected parallel to the expected shock location and located sufficiently outboard to cover the upstream spread of the captured shocks. All 60 mesh points were employed in the greatly reduced radial interval.

4. CONCLUDING REMARKS

In supersonic and hypersonic problems, the use of the PN/S method greatly simplified the mesh generation problem, reducing it from 3D to 2D. However, the severe stiffness of the problem together with the extremeness of the flow, particularly the hypersonic case, placed stringent quality requirements on the mesh. The treatment of shock waves is still a serious problem. Use of an adaptive mesh in the PN/S cross-flow plane will still require a mesh refinement in one transverse direction and in the marching direction. Shock fitting of the interior shocks does not offer an attractive alternative.

5. REFERENCES

1. Blom, G., Wai, J., and Yoshihara, H., "Calculations for a Generic Fighter at Supersonic High Lift Conditions", AGARD Symposium on Applied Computational Aerodynamics, Aix en Provence, April 1986.
2. Blom, G., Wai, J., and Yoshihara, H., "Hypersonic PN/S Calculations over a Generic Wing/Fuselage", Boeing Report BMAC Aero Note 86-111A, 1986. (Also presented at the SAE Aerospace Technology Conference, Long Beach, October 1986.)

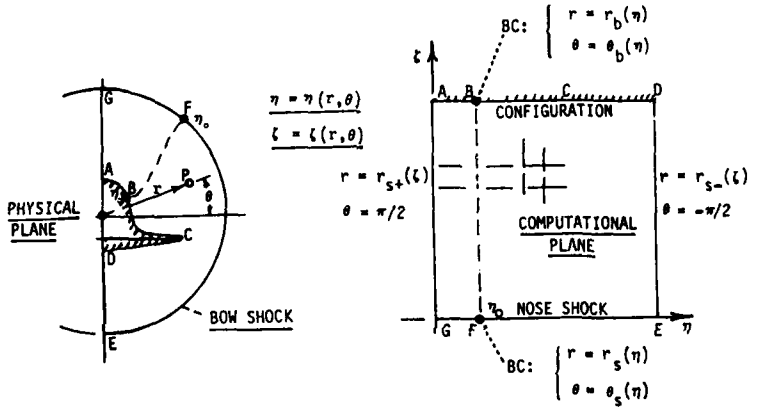


Figure 1. Transformation of the Marching Plane.

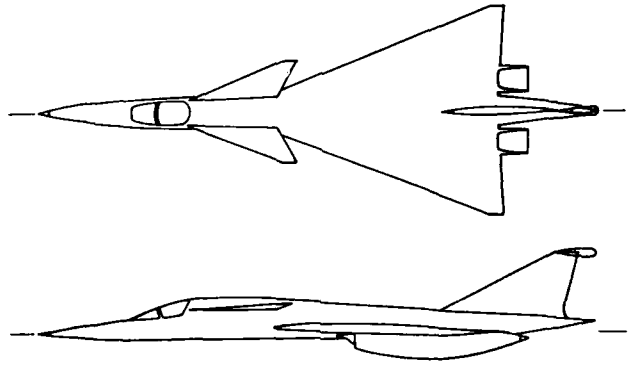


Figure 2. Model-350 Generic Fighter.

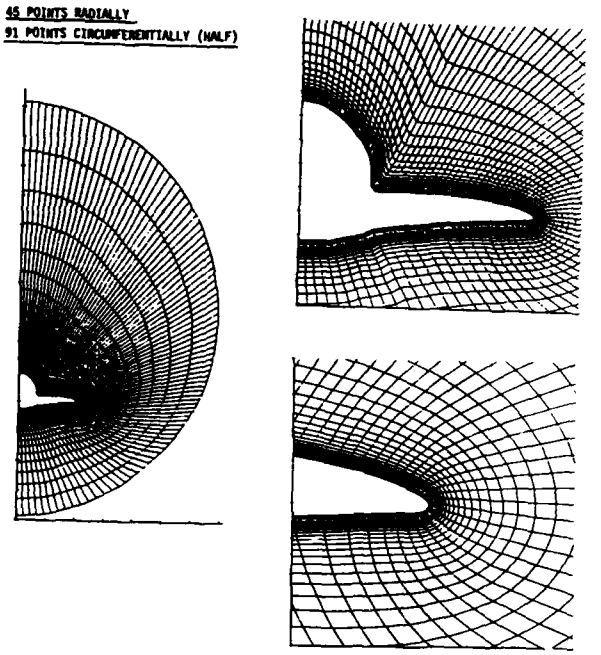


Figure 3. Model-350 Mesh - Wing Station.

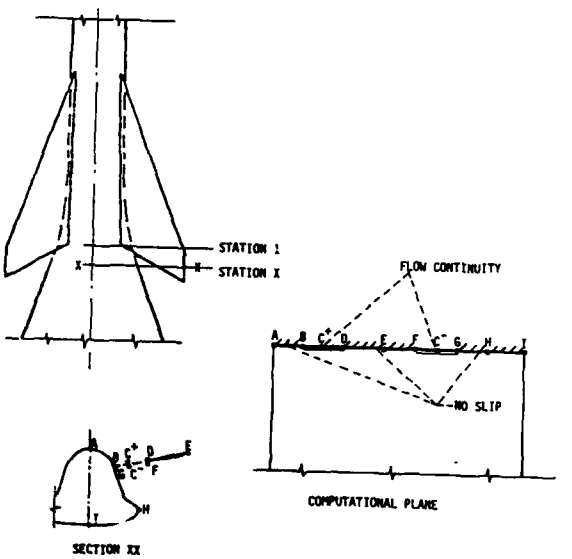


Figure 4. Mapping at the Canard Station - Model 350.

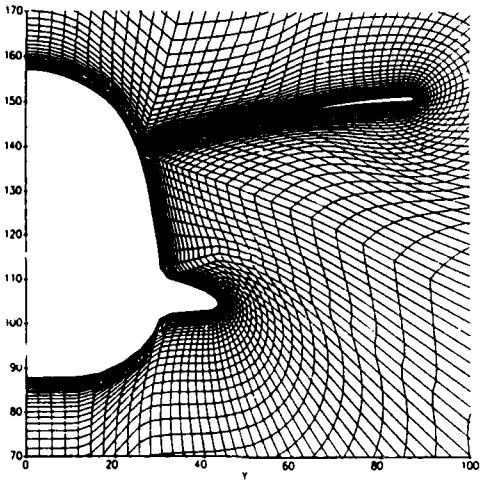


Figure 5. Mesh at Canard Station - Model-350.

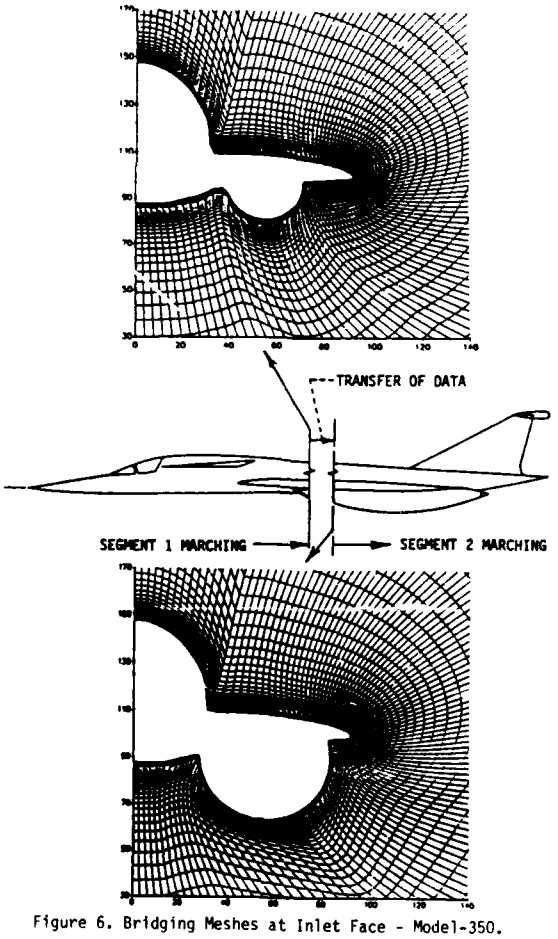


Figure 6. Bridging Meshes at Inlet Face - Model-350.

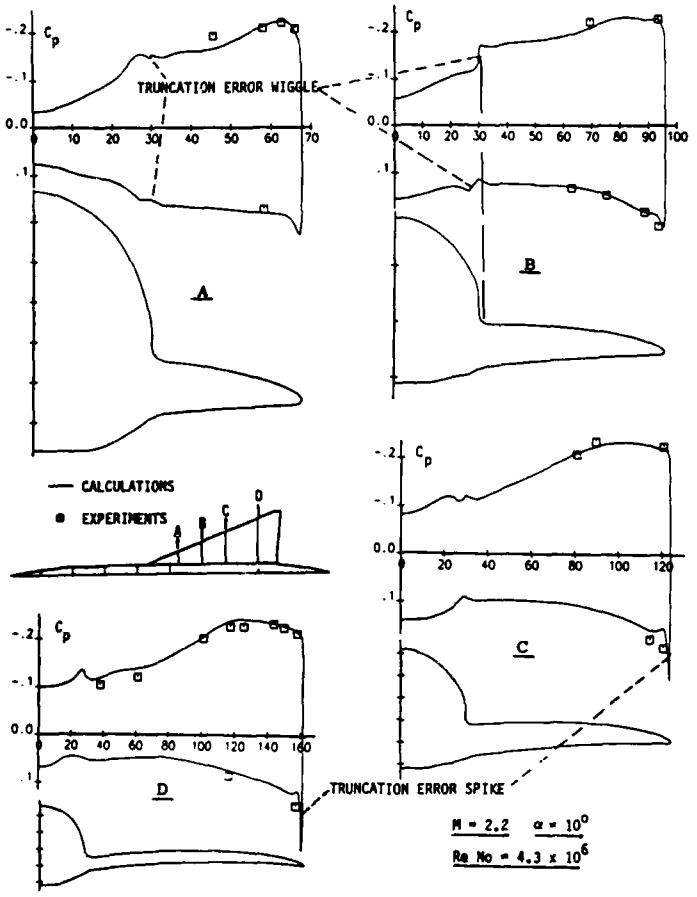


Figure 7. Truncation Error Pressure Spikes - Model-350.

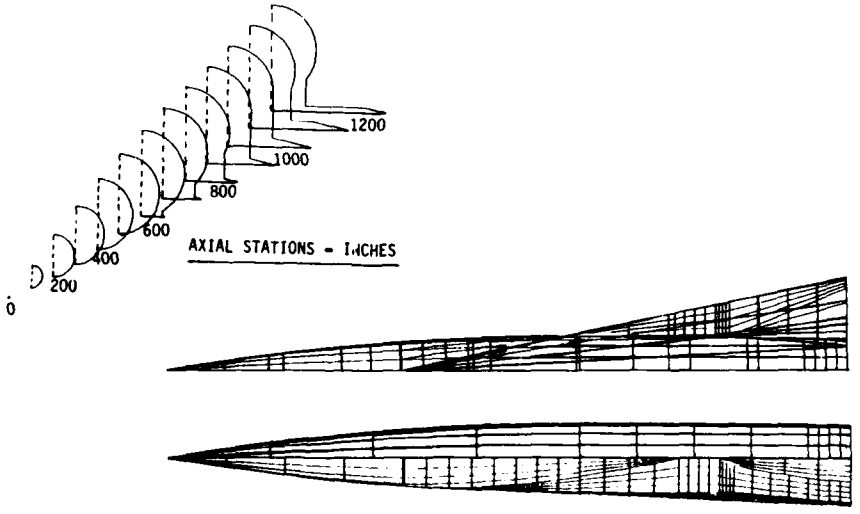


Figure 8. Hypersonic Generic Wing/Fuselage.

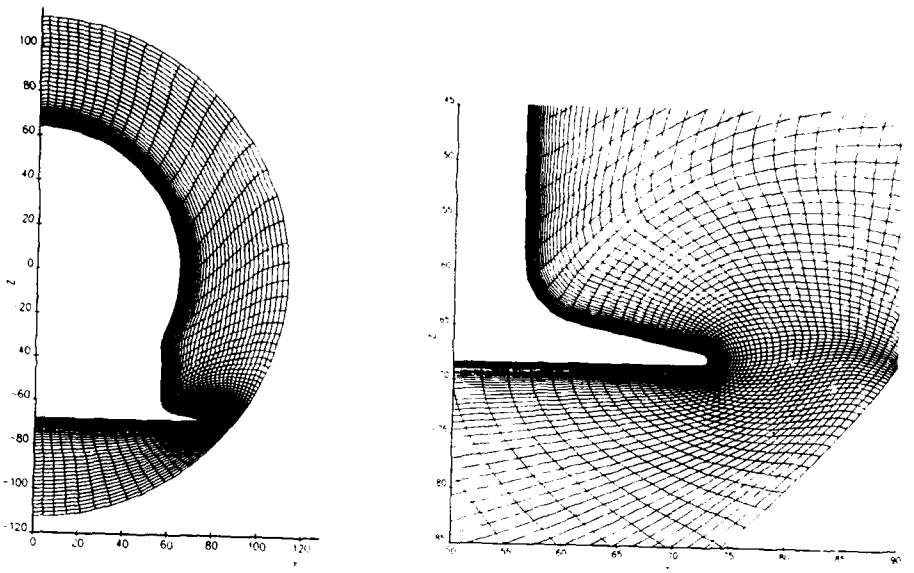


Figure 9. "Elliptically" Generated Mesh.

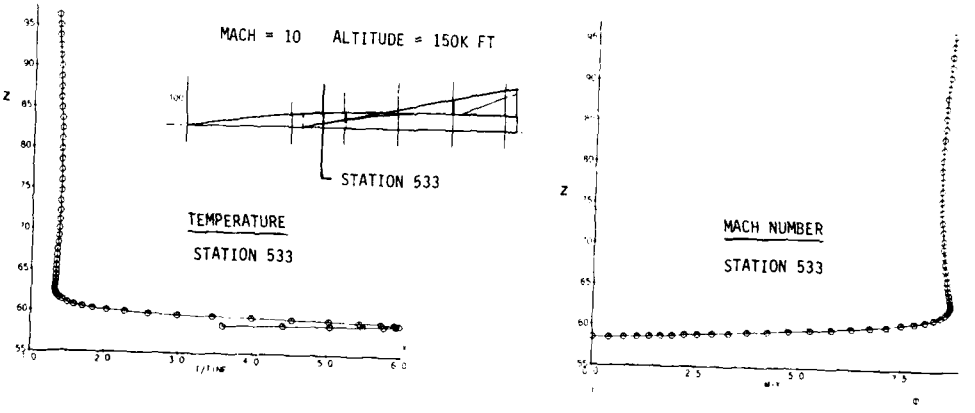


Figure 10. Temperature and Mach Number Profiles.

4.3 THREE-DIMENSIONAL ELLIPTIC GRID GENERATION FOR AN F-16

Reese L. Sorenson
Research Scientist
NASA/Ames Research Center
M/S 258-1
Moffett Field, CA 94035
USA

SUMMARY

A case history depicting the effort to generate a computational grid for the simulation of transonic flow about an F-16 aircraft at realistic flight conditions is presented. The flow-solver for which this grid is designed is a zonal one, using the Reynolds-averaged Navier-Stokes equations near the surface of the aircraft, and the Euler equations in regions removed from the aircraft. A body-conforming global grid, suitable for the Euler equations, is first generated using three-dimensional Poisson equations having inhomogeneous terms modeled after the two-dimensional GRAPE code. Regions of the global grid are then designated for zonal refinement as appropriate to accurately model the flow physics. Grid spacing suitable for solution of the Navier-Stokes equations is generated in the refinement zones by simple subdivision of the given coarse grid intervals. That grid generation project is described, with particular emphasis on the global coarse grid.

INTRODUCTION

The F-16 is widely employed in the US and NATO air forces, and that is the first of several reasons for the choice of that aircraft in this simulation. Secondly, many of its design features, such as leading-edge extensions or strakes, appear on fighter aircraft currently under development. Additionally, a great body of wind-tunnel data for that aircraft already exists. Lastly, Reznick[1] reports that there is work under way at present to expand the F-16's flight envelope to allow higher angles of attack. The propulsion system is capable of sustaining such attitudes, but certain undesirable flight characteristics prevent it. It is thought that computer simulation could aid that effort.

Zonal approaches have the advantage that geometrically complex flow fields such as this one can be simulated by zones which, by themselves, are geometrically simple.[2] Another advantage is that different flow-modeling equation sets can be used in different zones, as are appropriate to local flow conditions. This feature makes it possible to obtain solutions within a reasonable amount of computer time. The alternative, solving the Navier-Stokes equations everywhere in the field, would be computationally prohibitive. Further, in a zonal approach only the largest single zone need fit into the computer's high-speed memory at any one time. Here again, the alternative of putting the entire problem into the computer all at once would be prohibitive even for most modern computers. Also, the grid can be modified locally, for example decreasing the spacing according to the needs of the flow solver, without perturbing other parts of the grid.

GRID TOPOLOGY

A cylindrical grid topology about the fuselage is used because of its ability to treat a body with a sharp nose, as illustrated in Fig. 1. This topology also will facilitate the later addition of a grid zone for the exhaust plume. This fuselage grid is of the H-type as seen from the side or from the top. It appears to be of the O-type when viewed from the front or rear, thus giving rise to the terminology H-O-type as describing this cylindrical grid.

Further examination of Fig. 1 shows the grid above and below the wing, seen from the front, to be of the H-type, with the wing in a slit. The main advantages to the use of H-type topology in this coordinate surface are the ease with which it mates with the cylindrical fuselage grid, and its ability to provide an adequate number of points in the far field outboard of the tip.

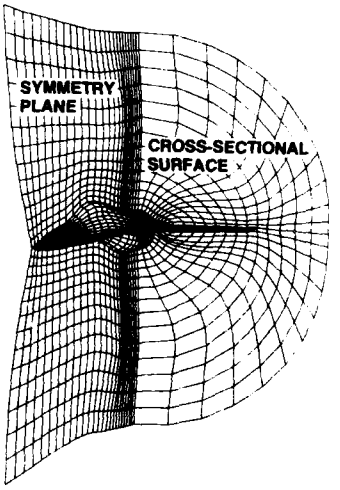
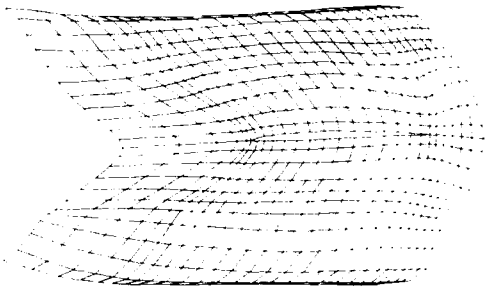
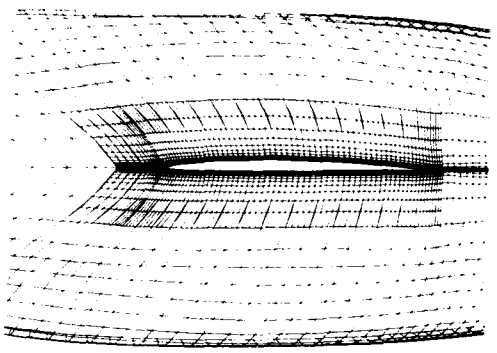


Figure 1. Front-Quarter view of F-16, showing fuselage, symmetry-plane, and wing.

A cross-sectional view taken normal to the span direction, shown in Fig. 2a, reveals an airfoil section with the surrounding grid being of the



a) Coarse global grid only.



b) Refinement zones added.

Figure 2. Grid surface normal to span.

H-type. This grid type could potentially waste many points upstream and downstream of the wing when the wing is embedded in a refinement zone, but the solution to this problem, shown in Fig. 2b, is to terminate that zone just upstream and downstream of the wing. Another disadvantage of this approach is that certain types of flow-solvers using H-type meshes have difficulties resolving a blunt leading-edge. But the leading-edge of the F-16 wing is very sharp, so this problem is minimized. The overriding advantage of this type of grid is the ease with which it can be mated to the fuselage grid. If the grid were of the O- or C-type in this direction, great difficulties would be encountered in mating to the fuselage, and in achieving adequate resolution upstream of the wing. Since the wing grid appears to be of the H-type when viewed in both span-normal and flow-normal directions, its topology is referred to as H-H-type.

When designing a mesh for a complex configuration such as this, it is difficult to quantify the effect of the chosen topolology on the serious matter of putting an adequate number of points where they are desired, principally near the fuselage- and wing-surfaces, while restricting the total number of points. Holst and Thomas[3] have attempted to clarify this matter with their Mesh Efficiency Ratio (MER). Reznick in Ref. 1 calculates that the MER has very favorable values for the type of grid topology used here, H-O on the fuselage and H-H for the wing. The efficiency of the mesh is important, since this flow solver challenges the speed and storage capabilities of even the most powerful modern computers.

This flow simulation was first performed for a simplified version of the aircraft, consisting of the fuselage, strake, and main wing only, with the inlet fared over.[4] Later addition of the inlet provided a perplexing topological challenge, as illustrated in Fig. 3a. The solution[5] chosen for this work was to introduce two warped wedge-shaped zones, shown in Fig. 3b. One wedge has its edge emanating from the lower side and bottom of the fuselage, upstream of the inlet, and its base at the face of the inlet. The other warped wedge-shaped zone nestles in the diverter region, between the top of the inlet and the bottom of the fuselage. The global grid wraps around the fuselage having these two zonal grids already attached.

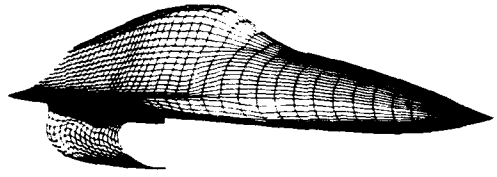
GLOBAL GRID GENERATION

The present approach to the generation of the coarse global grid follows the two-dimensional grid generation program* for airfoils of Sorenson and Steger.[6,7] This elliptic approach, with inhomogeneous terms that automatically control grid cell height and skewness at boundaries, was later extended to three dimensions for simple shapes,[8] and recently for realistic fighter aircraft.[9,10] This elliptic grid generation method has been seen, in both two- and three-dimensions, to be forgiving of surface slope discontinuities. Many surface slope discontinuities appear on this aircraft, e.g., at the edge of the strake, and the edges of the shelf aft of the main wing. Hence this grid generation method is appropriate for this problem. Another consideration is that the refinement grids, as seen in Fig. 2b, have some coordinate lines that are coincident with the course global grid. So for the refinement zone grids near the surface to be of reasonable size and not skewed, it necessary for the global grid to be well-controlled near the surface. The present grid generation method's ability to control cell height and skewness at boundary surfaces is therefore appropriate for this problem.

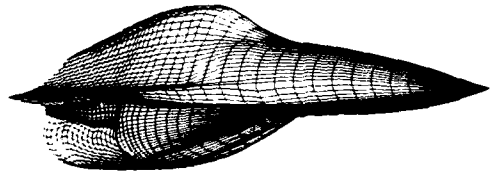
The program for generating grids about simple analytic shapes, reported in Ref. 8, was the starting point for this effort. The first step was to convert the program from

*Called GRAPE, an acronym for GRids about Airfoils using Poisson's Equation

spherical topology to cylindrical topology. The previous program with spherical topology used spatial variables rho,theta,phi. The new program is in cylindrical topology, but it uses cartesian spatial variables x,y,z. The cylindrical nature of the topology is imposed by the boundary conditions in physical space on the grid generation equations. The aircraft is assumed, for purposes of this study, to operate at zero yaw angle, hence the flow field can be assumed to be bilaterally symmetric about the vertical center plane. The flow-solver requires a one point reflection boundary at the symmetry plane, so that feature is included, both above and below the streamwise axis.



a) Right half of forebody.



b) Inlet and diverter zones added.

Figure 3. F-16 with inlet.

The next matter to be addressed was the treatment of the axis upstream of the nose of the aircraft. In the region of the fuselage, one face of the computational cube warps to conform to that fuselage. But upstream of the nose of the aircraft, that face collapses to the axis line. The Poisson equations can be solved, and thus a grid generated, in that case. However, the GRAPE grid generation program and its three- dimensional versions also solve side-condition equations to find the inhomogeneous terms which give the required control of cell size and skewness at boundary faces. Those side-condition equations become undefined when the face collapses to a point or a line.

Three possible solutions to this dilemma were considered. One would have been to just "turn off" the inhomogeneous terms there, i.e., set them to zero, leaving what would locally be the Laplace equations about the axis. It was expected, however, that such a grid would have been unacceptable. It was intended that the clustering effects on the body would strongly attract points to the body; to discontinuously release that attraction would have produced an unacceptable grid near the nose of the aircraft. The second possible approach would have been to fix the values of the inhomogeneous terms along the axis as being equal to (or some simple function of) the terms on the fuselage. This was rejected due to the perceived need to have terms at those places which really did constrain the grid as required there. The third solution, the one adopted, was suggested in private communication by by J. L. Steger and is illustrated in detail in Ref. 9. The axis, a line having a zero radius, was temporarily expanded to have a small positive radius using a simple analytic transformation. Thus the face which had collapsed to a line was re-expanded to resemble a narrow cylinder, or a "soda-straw." The grid was generated about this configuration, then shrunk back down to an axis by the inverse of that simple transformation in the radial direction. This artifice allowed the computation of inhomogeneous terms everywhere on that face of the computational cube.

The next challenge in this grid generation effort was to put the wing into its slit. Upstream of the leading-edge, outboard of the tip, and downstream of the trailing-edge, there is a planform grid surface, i.e., a grid surface in which the wing resides. But the wing, embedded in this planform surface, has two surfaces: upper and lower. So a fundamental data storage problem arises: how can one index refer to one grid surface off of the wing, and two surfaces on the wing? The solution used here is to make the planform surface to be "double-stored." There are two grid surfaces, denoted by two different indicies, which are coincident everywhere off of the wing. On the wing one of the two grid surfaces conforms to the upper surface of the wing, and one to the lower. To generate such a grid and use it in a flow-solver is tedious, but probably less so than other approaches. The principal coding *cavest* is to insert an "if test" when differencing across that double-stored surface, and appropriately increment the indicies.

The boundary condition arrangement in the grid generation to preserve the wing in its slit was made as simple as possible. Straightforward explicit boundary conditions were imposed on the upper and lower surfaces of the wing. It was felt that the ease of coding in this matter far outweighed any advantages which might have accrued from more sophisticated implicit boundary treatments.

However, in addition to instituting Dirichlet boundary conditions at the wing, it was necessary to add inhomogeneous terms to impose on the wing surfaces the same kind of control of cell height and skewness as is imposed on the fuselage. See Ref. 10 for a detailed treatment of those inhomogeneous terms, as well as the Poisson grid generation equations to which they are applied. Those terms are similar in form to the body's terms, but required some coding effort, since the wing surfaces are examples of a

different family of grid surfaces *vis-a-vis* the fuselage surface. The fuselage, in this effort, is a surface of constant computational variable zeta. The wing surfaces are surfaces of constant xi. Inhomogeneous terms were also added to cluster lines in the planform grid surface toward the leading-edge and trailing-edge of the wing. Those terms are degenerate two-dimensional versions of the terms on the body and wing surface, and are identical in form to the two-dimensional clustering terms in the GRAPE program.

The attempt was also made to add similar terms to cluster lines outboard of the tip, to make lines in that region to be near-orthogonal as viewed from above. That effort was not successful, however. It is a fundamental trait of elliptic grid generation methods, including this one, that grid lines tend toward a uniform distribution. Thus, if points on a boundary are tightly clustered in the direction tangent to that boundary, and lines emanate from those tightly clustered points, those lines will tend to repel each other. That mutual repelling action can be so intense that the lines deviate greatly from being orthogonal to the boundary, regardless of the presence or absence of clustering terms. This problem was encountered when attempting to add inhomogeneous terms at the wing tip for the purpose of controlling spanwise lines proceeding outboard from the tip; see Fig. 4. A grid solution could not be obtained; the grid equation convergence history became oscillatory. The resolution of this problem was to extend the wing outboard with zero thickness all the way to the outer boundary. Thus spanwise lines in that region were defined as part of the initial conditions to the grid generation, and they remained in place for all computational time.

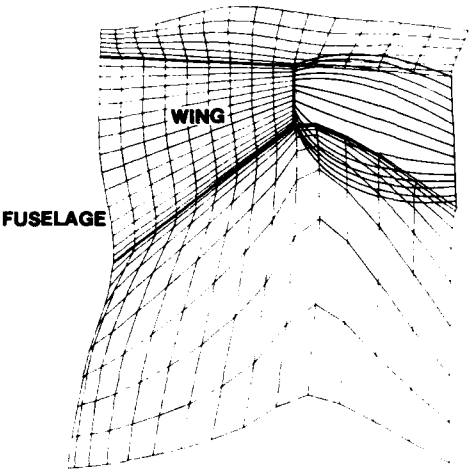


Figure 4. Sketch suggesting the tendency of grid lines emanating from closely-spaced boundary points to mutually repel, causing instability.

The next step in the process of applying the present grid generation technology to the F-16 flow simulation problem was to fit the surface of that aircraft, and to obtain a good distribution of body points. This proved to be a formidable task. The body definition was obtained in the form of tabulated points describing the surface. However, the definition of the airplane's shape so obtained was in several pieces, with different coordinate systems for each piece. Some of the pieces did not exactly fit together. Further complicating this situation was our intention to start with an F-16 simplified by a fared-over inlet and the deletion of the empenage, missile rails, and ventral fins. These modifications, having the purpose of simplifying the initial task of the flow-solver, made the body-fitting significantly more difficult. The body-fitting effort was performed by Edwards[11] on a Calma CAD/CAM system. While this device and its software aided in the smoothing and faring operations, certain deficiencies were found which made interpolating in some coordinate directions practically impossible. Thus those point distribution functions devolved to be part of the grid generator. These body-definition and distribution problems reached a zenith later, when attempting to restore the inlet. For various reasons having to do with the availability of personnel and machines, almost three man months were spent creating the wedge zones, a task which should have bordered on the trivial.

Distribution of body points is an area of concern, even under ideal circumstances. An elliptic grid generator is sensitive to not only the shape of the body, but the distribution of points on it. Consider a trace proceeding around the fuselage in a plane cutting it normal to the streamwise axis, with that trace proceeding across the edge of the strake. The body points on that trace must be clustered approaching the strake edge from both directions, with minimum tangential spacing immediately above and below the edge. (A two-dimensional analogy to this is that points must be clustered to the nose of a sharp-nosed airfoil when generating an O-type grid about it.) If these precautions are not taken, the elliptic grid generator will at best give a grid with odd angles at the edge, and at worst fail to converge.

Distribution of body points is likewise critical for the flow-solver as well. In the early months of this effort, computer storage limited the number of grid points in the chordwise direction on the wing, which in turn limited how fine the spacing could be in that direction at the leading-edge. Thus the first effort failed to resolve the shape of the leading-edge, leading to a failure to correlate with test data at high angles of attack. This problem was resolved in later efforts with NASA/Ames' new CRAY-2 computer, having 256 million words of high-speed memory.

The resulting coarse global grid, illustrated in Fig. 5, has 26 points around one

half of the fuselage, from bottom to top, including the symmetry plane and the reflection boundary surfaces. The grid has 55 points in the streamwise direction, including 5 upstream of the nose and 5 downstream of the jet-exhaust. Twenty points are used in the radial direction, giving 28,600 total points in the coarse grid. Fifteen points are used in the chordwise direction on the wing, with 10 spanwise stations. The minimum spacing normal to the fuselage surface, between the surface and the next point outward, was controlled by the GRAPE-type terms to be approximately 4 inches on the real aircraft. Approximately the same spacing was required on the upper and lower surfaces of the wing.

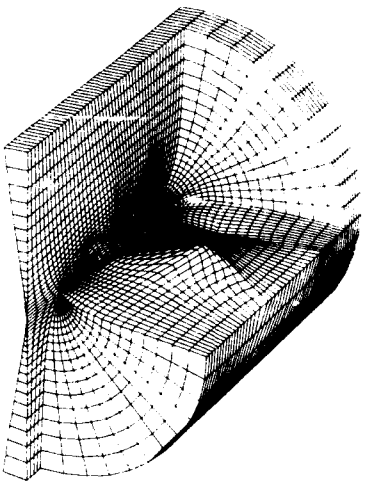


Figure 5. Cutaway view of coarse grid.

The differential equation solver used in the grid generation was point-SOR. While this method is admittedly one of the slower ones in use today, it was chosen for its simplicity and ease of coding. The placing of the wing in its slit, for example, would have been a much more complicated task with a more sophisticated equation solver method, such as ADI. With no effort at vectorization, this FORTRAN code produced coarse grids in aprox. 100 iterations at aprox. one second per iteration on a CRAY XMP 4/8 computer.

GENERATION OF REFINEMENT GRID ZONES

Refinement zones were specified to provide viscous clustering in the direction normal to the surface wherever it was expected that viscous effects would be significant. Grid points added in the surface-normal direction in refinement zones were located by a bisection iteration scheme applied to arclengths between consecutive points, for the purpose of enforcing a constant ratio between the lengths of successive grid intervals. Grid points were added in surface-tangent directions by dividing existing coarse grid intervals into simple fractions of their previous length, e.g., by halving, quartering, etc. Refinement zones are seen in Fig. 2b, a view of a spanwise-normal grid surface showing refinement zones surrounding the wing, and in Fig. 6, a streamwise-normal grid surface near the wing. Details of the generation of the refinement zones are described and illustrated on pages 22-31 of Ref. 1. A total of 18 zones were used in this case with the fared-over inlet, having in them a total of 304,134 points. Flow-field solutions were obtained using this grid, and are described in Ref. 4. The grid for the case with inlet, including 19 zones, contains a total of 384,094 points. Flow-field solutions are described in Ref. 5.

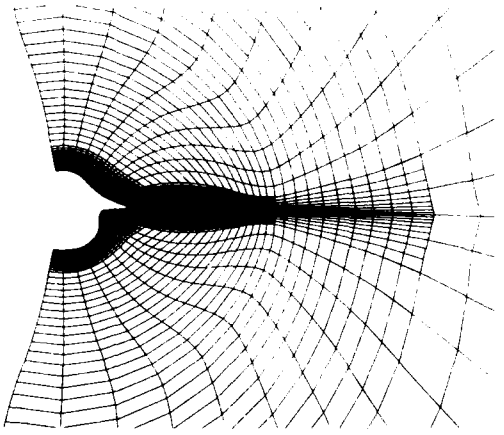


Figure 6. Streamwise-normal grid surface near wing, showing refinement zones.

CONCLUSIONS AND FUTURE PLANS

The first and most obvious conclusion to be drawn here is that the above method works. Grids were generated, and flow-solutions to this very difficult problem were obtained, and they did agree with test data. See Refs. 4 and 5.

But the call for this paper requested "...a description of what you did and what worked....some details of what were the most time consuming tasks and what difficulties you had to overcome....a brief written tutorial on how you generated the grid for your application." In accordance with that request the following observations are made.

The most problematical part of this grid generation project was obtaining an adequate fitting of the body surface, and distributing points on it, i.e., obtaining an adequate surface grid. There were several contributing factors to this matter, including inadequacies and unreliabilities in the CAD/CAM software, and difficulties in obtaining time on the unique hardware on which it must run. Also contributing here was the marginal suitability of the "raw" body

definition which was input to the CAD/CAM, although this problem may be traced to incompatibilities between the different CAD/CAM systems here and at the airframe manufacturer. An effort is underway as of this writing to adapt the PATRAN software to this task, but results have not yet been obtained.

But a more fundamental problem here may be the way surface-fitting is viewed, relative to other parts of the field of Computational Fluid Dynamics (CFD). Body-fitting is not seen in some quarters as a "glamorous" career direction. Those performing these tasks sometimes move on to other regions of the field, leaving a lack of continuity, a lack of the easy expertise which comes with experience. A solution may be to recognize body-fitting for what it is: a major pacing item in modern application-ally-oriented CFD, and to accord it the respect which it deserves. Another approach might be to have an easy-to-use "turnkey" CAD system, one which is *oriented to CFD applications,* and could be easily used as a tool by any CFD researcher to obtain superior body fitting.

Some significant grid generation technology has been developed in this work, specifically an extension of GRAPE to three-dimensions and the application of it to zonal approaches. This author has had several requests for this grid generator from other CFD researchers, and so an effort is underway to package it for export. This requires complete re-coding, in FORTRAN, to produce a program which is neat, modular, robust, well-documented, and easily applicable to a wide variety of cases. The program will allow the flow-field to be broken-up into a large number of zones, and it will be capable of solving the grid-generation equations in all zones simultaneously, with information passing between zones. It will be possible to impose near-orthogonality and control of surface-normal spacing at all six faces of each zone. Single-dimension addressing will be used, facilitating the generation of zones having a great latitude in their sizes (e.g., it will be possible to generate one zone with dimensions 10 x 10 x 100, a second zone being 10 x 100 x 10, and a third zone that is 100 x 10 x 10, all with the same program which does *not* have dimensions 100 x 100 x 100). The program will not, however, have any capabilities for fitting body surfaces or distributing points on them; it has been (and continues to be) the philosophy of the GRAPE grid generator that surface-fitting is a formidable problem in itself, and that the surface-grid is a boundary condition which should be an input to the grid generator.

REFERENCES

[1]Reznick, S. G., "Transonic Navier-Stokes Computations of Strake-Generated Vortex Interactions for a Fighter-Like Configuration," Ph.D. dissertation submitted to Stanford University, Dec. 1986; also to appear as NASA TM.

[2]Holst, T. L., Gundy, K. L., Flores, J., Chaderjian, N. M., Kaynak, U., and Thomas, S. D., "Numerical Solution of Transonic Wing Flows Using an Euler/Navier-Stokes Zonal Approach," AIAA paper 85-1640, July 1985; also accepted for publication in J. Aircraft.

[3]Holst, T. L., and Thomas, S. D., "Numerical Solution of Transonic Wing Flow-fields," AIAA Journal, Vol. 21, no. 6., June 1983, pp. 863-870.

[4]Flores, J., Reznick, S. G., Holst, T. L., and Gundy, K., "Transonic Navier-Stokes Solutions for a Fighter-Like Configuration," AIAA paper 87-0032, Jan. 1987.

[5]Flores, J., Chaderjian, N. M., and Sorenson, R. L., " Simulation of Transonic Viscous Flow Over a Fighter-Like Configuration With Inlet," accepted for presentation at AIAA 19th Fluid Dynamics, Plasma Dynamics and Laser Conference, June 8-10, 1987, Honolulu, Hawaii.

[6]Sorenson, R. L., "A Computer Program to Generate Two-Dimensional Grids about Airfoils and Other Shapes by the Use of Poisson's Equation," NASA TM-81198, May 1980.

[7]Steger, J. L., and Sorenson, R. L., "Automatic Mesh-Point Clustering Near a Boundary in Grid Generation with Elliptic Partial Differential Equations," J. Comp. Phys., Vol. 33, no. 3, pp. 405-410.

[8]Sorenson, R. L., and Steger, J. L., "Grid Generation in Three Dimensions by Poisson Equations with Control of Cell Size and Skewness At Boundary Surfaces," Advances in Grid Generation, Ghia, K. N., and Ghia, U., eds., ASME, New York, 1983, pp. 181-188.

[9]Sorenson, R. L., "Three-Dimensional Elliptic Grid Generation About Fighter Aircraft for Zonal Finite-Difference Computations," AIAA paper 86-0429, Jan. 1986.

[10]Sorenson, R. L., "Elliptic Generation of Composite Three-Dimensional Grids About Realistic Aircraft," Numerical Grid Generation in Computational Fluid Dynamics, Häuser, J. and Taylor, C., eds., Pineridge Press, Swansea, U. K., 1986, pp. 353-371; also as NASA TM-88240, March 1986.

[11]Edwards, T. A., "Definition and Verification of a Complex Aircraft for Aero-dynamic Calculations," AIAA paper 86-0431, Jan. 1986.

4.4 COMPONENT ADAPTIVE GRID GENERATION FOR AIRCRAFT CONFIGURATIONS

by

N.P. WEATHERILL

J.A. SHAW

Aircraft Research Association Ltd., Manton Lane, Bedford, England

INTRODUCTION

As numerical algorithms for the solution of the Full Potential and Euler equations mature there is an increasing demand to simulate the flow over complex aerodynamic shapes. However, of the many problems in engineering and the physical sciences which involve grid generation, the shapes and length scales encountered in aerodynamics are some of the most varied. The generation of a suitable set of points around an aircraft shape, therefore, affords a significant technical challenge and one which is currently being pursued by many groups.

The concept of the grid generation technique adopted at the Aircraft Research Association was originally proposed by Forsey as discussed in Ref.1. The development of a general grid generation method applicable to a wide range of aircraft configurations, involves a block decomposition of the flow domain with grid points within each block computed by solution of a set of elliptic partial differential equations. This method enables grid structures to be constructed which are compatible with each separate component in a configuration while maintaining a globally smooth grid. This component adaptive grid generation technique has been applied to a variety of configurations and details of the method have been given elsewhere[2,3,4,5]. It suffices, therefore, to give only a brief outline of our approach.

The method utilises the basic concept of block structured grids. The flow domain is subdivided into a set of non-overlapping blocks. The arrangement of the blocks defines the grid structure or grid topology which is appropriate for the geometrical configuration. The block subdivision is performed automatically by a block decomposition algorithm. Each block is chosen to be topologically equivalent to a cuboid in that it has six faces and eight corners and can, therefore, in principle, be mapped into a unit cube in computational space without change in topological structure. Cartesian grids in the unit cubes in computational space map to curvilinear grids in physical space. Many faces within the block structure are boundaries between blocks in the interior of the flow domain and as such are purely notional boundaries which have no physical significance. At such boundaries a continuity condition can be imposed which ensures grid lines pass smoothly through the interface of two adjacent blocks. Following the ideas of Thompson, Thames and Mastin[6], a set of elliptic partial differential equations have been used to generate grid point coordinates within each block. These equations can be written

$$g^{ij} X_{\xi^i \xi^j} = -p^i X_{\xi^i} \qquad (1)$$

where g^{ij} are the metric terms, X the grid point coordinates and ξ^i the computational coordinates with the tensor notation ij taking values of 1,2 and 3. The source terms p^i are used to control the positioning of grid points and their form is computed using the ideas of Thomas and Middlecoff[7]. The continuity condition at block faces is applied by defining a computational molecule for points on the faces which is compatible with the finite difference solution of the elliptic equations (1).

Following the ideas of Coons[8] the surface of each component of a configuration is modelled by a network of parametric bi-cubic patches. Any patch can be described by the matrix equations

$$AMB^{-1} = X$$

where $X = (x,y,z)$, $A = (s^3,s^2,s,1)$, $B = (t^3,t^2,t,1)$ and M is a matrix containing the parametric derivatives of X and some blending functions. Grids on the surface of a geometry are computed in the parametric space (s,t) using the equivalent two-dimensional form of equation (1).

Here we propose to discuss the application of these techniques to wing-body-canard geometries. These configurations are sufficiently complicated to highlight the difficulties inherent to grid generation and provide good test cases on which a more detailed discussion of our approach can be based.

TEST CASE

The test case geometries are shown in Figures 1 and 2. The two afford interesting geometrical contrasts and represent two typical configurations which might be confronted in the Aerospace industry. Geometry A, shown in Figure 1, has a swept forward wing with the canard position at the body side approximately a canard chord forward of the wing. The canard elevation is higher than that of the wing.

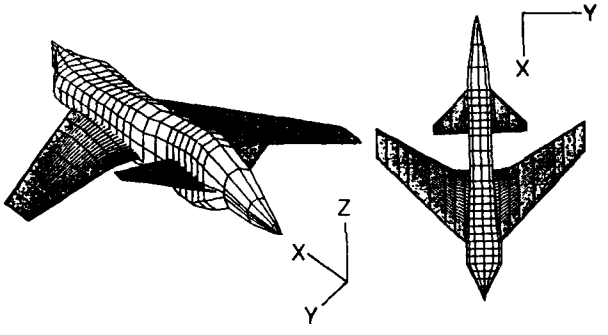


FIGURE 1. GEOMETRY A. SWEPT FORWARD WING WITH CANARD AND BODY.

Geometry B is shown in Figure 2 where it should be noted that the body has been extended far downstream to avoid problems with afterbody effects. At the bodyside, the x-value of the trailing edge of the canard is approximately equal to the x-value of the leading edge of the wing. Due to the sweepback of the leading edge of the wing, with respect to the trailing edge of the canard, the x-value of the trailing edge of the canard tip is several canard tip chords upstream of the leading edge of the wing. This spanwise variation of the relative positions of the wing and canard leads to a conflict in the appropriate chordwise topology for the body side grid and the canard tip grid. The elevation of the canard is above that of the wing.

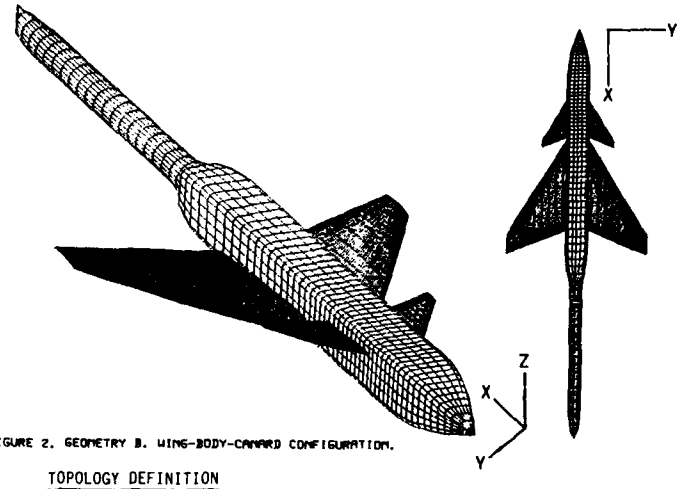


FIGURE 2. GEOMETRY B. WING-BODY-CANARD CONFIGURATION.

TOPOLOGY DEFINITION

In principle, the multiblock method described above allows a wide range of grid structures to be defined for a given configuration. However, the problems associated with grid control are strongly influenced by the choice of grid structure for a geometry. A wise choice of grid topology, which utilises the properties of the elliptic equations (1) used to generate the grid, can ease the requirements on the grid control technique. However, an inappropriate grid structure can lead to unacceptable demands on any coordinate system control method.

In addition to grid control requirements, the specification of any suitable topology can be problematic. The mechanics of defining a grid structure within the multiblock framework involves specifying, for each block, the number of grid points in each computational coordinate direction, and for each face the appropriate boundary condition type for the elliptic grid generation equations. For the continuity boundary condition it is necessary to specify the adjacent block, the appropriate adjacent face of that block and the orientation between the coordinate axes fixed in each block. The specification of this information although straightforward for a simple grid structure is time consuming and tedious. If one aspect of a configuration is changed, for example, the canard moved a small distance relative to the wing, then this may involve a significant modification to the topology.

It is clear from these brief comments that the definition of a suitable topology for a given configuration requires expert knowledge. A grid generation technique, which is to be used by non-CFD specialists, must overcome the problem of topology construction. To this end, an automatic topology generator has been developed which, given a configuration, subdivides the flow domain into a set of blocks, the arrangement of which is consistent with a component adaptive grid structure.

The ideas behind the automatic block decomposition algorithm are best illustrated in two dimensions. Consider an aerofoil in a finite two dimensional domain. In computational space (ξ,η) the aerofoil coordinates map to a cut of constant η. In a block structured domain this cut maps to a side of a rectangular block. Assuming a unique boundary condition type for each side, a block decomposition of the domain would result in six regions, as shown in Figure 3a. Now introduce three additional cuts in the computational domain; two cuts of constant η, one above and one below the aerofoil and a third cut of constant ξ - upstream of the aerofoil. The resulting block decomposition is shown in Figure 3b. It is now possible to make a small change in the block structure to construct a grid topology in which a C-grid is locally embedded around the aerofoil within a global H-grid. Such a transformation is shown in Figure 3c.

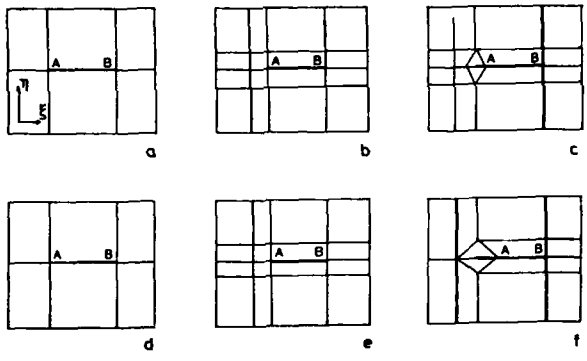


FIGURE 3. BLOCK DECOMPOSITION FOR AN AEROFOIL.

Clearly the transformation performed around point A in Figure 3c could be applied around B to give a polar structure around the aerofoil. Other transformations are possible which give rise to other block structures and different singularities in the grid. One such example is shown in Figures 3d, e and f wherein the block decomposition gives rise to a six point singularity ahead of the aerofoil section. Experience in generating grids with singular points indicates that although the position of singularities is not easy to control, two five point singularities away from the aerofoil are preferred to a six point singularity positioned just ahead of the leading edge. The type of transformation shown between Figure 3a and 3c has been adopted in the automatic topology generator.

The arguments presented here in two dimensions are applicable to three-dimensional shapes like a wing, pylon, tail, body etc. In such cases the local block structure around each component is pseudo three-dimensional in the sense that the locally adapted grid structures shown in Figure 3c are repeated along the component. At the termination of the component the same grid structure continues but is constructed around a degenerate form of the component. For example, a locally embedded C grid around a wing is continued outboard of the tip where it is constructed around an imaginary extension of

the wing which has zero thickness. For a locally embedded polar grid around a body this degenerates into a polar grid encircling a singular line upstream of the body nose.

AUTOMATIC TOPOLOGY GENERATOR

Details of the configuration are input to the topology generator by means of a schematic of the geometry. Each component of a configuration is input as a plane of constant x,y or z and given the flow domain to be within the cube [0÷1000, 0÷1000, 0÷1000],it is straightforward to generate the appropriate block structure for a Cartesian grid given the constraint of one boundary condition type per block face. Following the ideas sketched in two dimensions, a first step to the construction of a component adaptive grid topology is the addition of planar cuts in the domain. For a given isolated component the local embedding of a C or O structure is reasonably straightforward. The key to extending such an approach to multiple components is to ensure that the new blocks introduced to provide the C and O structures never map to other new blocks for different components. In other words, the introduction of the new blocks must always be performed in a locally Cartesian block structure.

In practice, the use of the algorithm is straightforward. Given the geometry of Figure 1 it is clear that it is possible to construct several topologies within the class of topologies generated by the automatic approach. The freedom to construct different grid structures rests with the specification of the schematic. A study of the geometrical features could lead to the definition of the following schematic.

Body: (300,0,300), (700,0,300), (300,0,700), (700,0,700)
Wing: (500,0,500), (600,0,500), (500,400,500), (600,400,500)
Canard: (400,0,550), (450,0,550), (400,200,550),(450,200,550)

In this form the canard is upstream of the wing and lies above the wing elevation. Such a schematic (schematic A) would lead to a C-structure around the canard which continued above the wing. An outline of the expected grid structure at the body side is given in Figure 4.

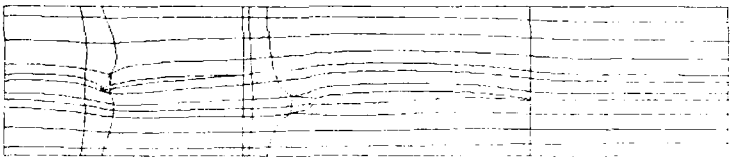


FIGURE 4. GRID STRUCTURE ON THE BODY DERIVED FROM GEOMETRICAL SCHEMATIC A.

In contrast, it would also be appropriate for geometry A to define a schematic in which the elevation (i.e. z coordinate value) of both wing and canard were the same. In this case the schematic for the canard could be redefined to be

Canard: (400,0,500), (450,0,500), (400,200,500), (450,200,500)

This schematic,(schematic B), would then lead to a grid structure on the body side as indicated in Figure 5. Both topologies are sensible for the configuration and the better of the two can only be determined by viewing the grids generated by the two approaches.

Having defined a schematic, the topology generator, which is executed interactively, performs the necessary planar cuts ready for the embedding of the C and O grid structures. The user is prompted for the block dimensions, which, since at this stage the block structure is Cartesian, requires the specification of IB+JB+KB values, where IB,JB and KB are the number of blocks in each of the coordinate directions. The transformations are applied to produce the final topology and the new blocks are assigned dimensions consistent with the existing structure. For the first schematic the total number of blocks in the construction is 430 (IB=9,JB=4,KB=11) and in the second 322 (IB=9,JB=4,KB=8). The topology generator also outputs auxiliary information for the post-processing of grids and flowfield solutions. The algorithm also outputs information which is used by the grid generators to assist with grid control.

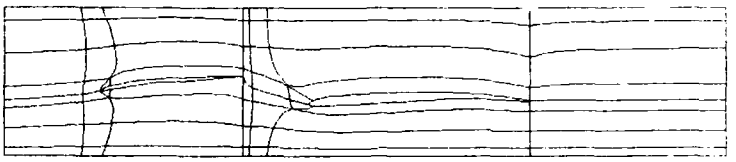


FIGURE 5. GRID STRUCTURE ON THE BODY DERIVED FROM GEOMETRICAL SCHEMATIC B.

The grid structure on the surface of each component and on the outer boundary is readily derived from the three-dimensional grid topology. The outer boundary is constructed from a number of different components. These components, which when combined form a rectangular box, are defined to be consistent with the field topology. For example, the planar cut in the topology at the wing tip extends to the outer boundary and forms the intersection between two parts of the boundary. It is consistent with the spanwise grid topology to set this intersection line to the y coordinate of the wing tip. A similar procedure is adopted on the cut at the canard tip. Each component of the outer boundary is modelled in a similar manner to a component of the configuration and the grids are generated in parametric coordinates.

SURFACE GRIDS

The grids on the surface of each component are generated in parametric coordinates and in an order which ensure that grid properties of one component can, if necessary, be used to ensure a consistent grid on another component. The grid topology for the body, consistent with the field topology generated from schematic A of the wing-body-canard configuration geometry A, is given in Figure 6.

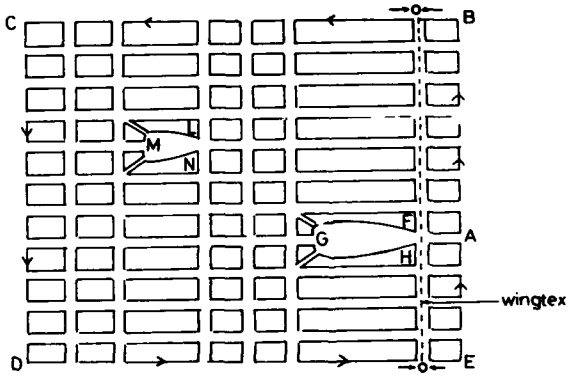


FIGURE 6. BLOCK STRUCTURE FOR SURFACE GRID ON BODY USING SCHEMATIC A.

The contour path ABCDEA represents the body/plane of symmetry intersection line and the paths FGH, LMN the intersections between the wing/body and canard/body, respectively. To generate the grid in this block structured domain using the elliptic equations (1) the necessary boundary data must be defined on these paths. The appropriate point distribution is of considerable importance since, using the Thomas and Middlecoff approach to compute the control function p^1, it effectively determines the quality of the grid in the interior of the domain. Grid points on the paths FGH and LMN are computed from a geometrical intersection routine which ensures that appropriate grid clustering occurs at the leading and trailing edges. To ensure a suitable distribution of points along the path ABCDEA, information from the automatic topology generator is utilised. Descriptors which are associated with features of the geometry and topology paths are assigned to particular blocks and edges. For example, the descriptor wingtex is a path of constant x associated with the trailing edge of the wing. As the contour path along ABCDEA is prescribed, the point distribution routine within the surface grid generator examines any topology path descriptor which crosses the contour path in a normal direction. When two such paths cross, grid points are attracted to both sides of the intersection. Additional attraction and repulsion of grid points occurs where corners in the contour path are detected. In this way appropriate grid point

clustering occurs on the path ABCDEA which is consistent with the configuration and the topology. In schematic form the idea is illustrated in Figure 6. This approach, outlined for the body, is applicable to the generation of all surface grids.

Figure 7 shows a grid in the parametric coordinates for the body of geometry A using schematic A for the topology. Grid point clustering, so carefully constructed on the boundaries, can be seen to influence the interior grid points. The grid in physical coordinates along with the component grids on the wing and canard, is illustrated in Figure 8. The nose of the body is reasonably well defined using the polar structure in the field topology and the inserts show the locally embedded C grids around the wing and canard. Figure 9 illustrates the effect of grid point clustering on the component parts of the outer boundary. This proves essential in achieving a good quality grid in the field.

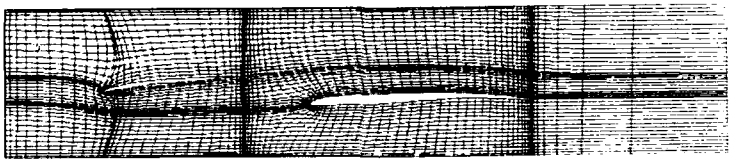


FIGURE 7. SURFACE GRID IN PARAMETRIC COORDINATES FOR THE BODY OF GEOMETRY A USING TOPOLOGY DERIVED FROM SCHEMATIC A

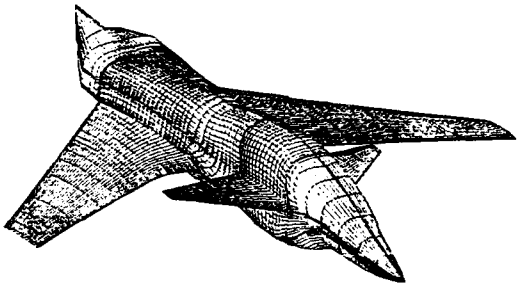


FIGURE 8. GRID ON THE SURFACE OF GEOMETRY A SHOWING THE GRID STRUCTURE OBTAINED FROM SCHEMATIC A.

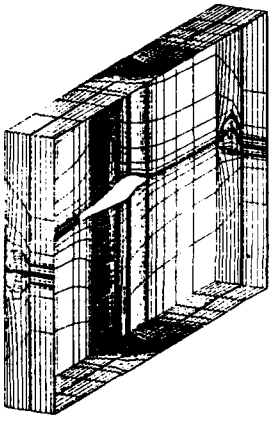


FIGURE 9. GRID ON THE OUTER BOUNDARY OF THE FLOW DOMAIN.

This method of grid point distribution is applicable to all grids generated from the automatic topology generator. To illustrate its use on the topology generated from schematic B, Figure 10 shows the grid on the surface of the configuration. The topological differences between grid structures in Figure 8 and 10 are evident. Both grids are of a high quality, but perhaps

the grid structure in Figure 8 results in slightly less skewed cells in the region between the wing and canard. In both cases, the forebody is well defined.

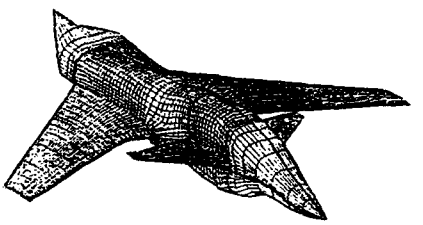


FIGURE 10. GRID ON THE SURFACE OF GEOMETRY A SHOWING THE GRID STRUCTURE OBTAINED FROM SCHEMATIC B.

FIELD GRID

Once the grids on the surface of the configuration, the plane of symmetry and the outer boundary have been generated, the grid points in the field are derived by solution of equation (1). The source functions p^i on the boundaries are computed from the fixed boundary data and interpolated through the field ensuring that the $p^i (i = 1,2,3)$ are consistent on block faces and edges.

Grid control may be enhanced by fixing some internal block boundaries but this greatly increases the labour of grid generation. Sections of a field grid generated using the topology of schematic A are shown in Figure 11. The component adaptive nature of the grid is evident with locally embedded C grids around the wing and canard and a polar grid around the body. The grid point distribution in the field can be modified by an appropriate choice of parameters which modify the grid control functions p^i in particular regions of the domain.

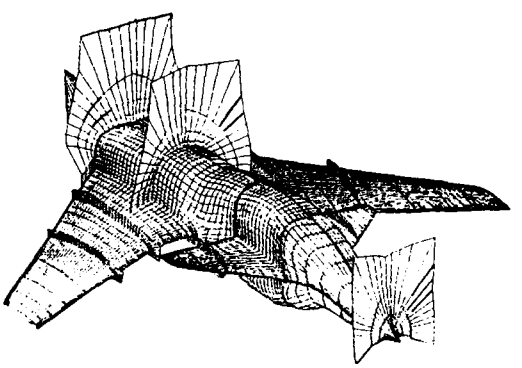


FIGURE 11. FIELD GRID SECTIONS HIGHLIGHTING THE COMPONENT ADAPTIVE TOPOLOGY STRUCTURE.

As a final illustration of the power of our approach we will apply the method to geometry B. As was already noted, the spacing between the canard and the wing at the body side and the canard tip leads to a conflict in the appropriate choice of topology. The geometry at the body side would indicate a suitable schematic (schematic C) for the topology to be

Body: (300,0,300), (700,0,300), (300,0,700), (700,0,700)
Wing: (500,0,500), (600,0,500), (500,400,500), (600,400,500)
Canard: (400,0,550), (500,0,550), (400,200,550), (500,200,550)

This implies that the trailing edge of the canard has the same x value as the leading edge of the wing. A schematic of the grid on the body side for the topology is shown in Figure 12. However, if the position of the canard with respect to the wing is noted at the canard tip, it would appear sensible to define the canard trailing edge to be forward of the leading edge of the wing. This would lead to the schematic A and the appropriate grid structure for geometry B is given in Figure 13.

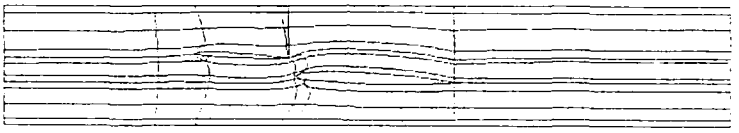


FIGURE 12. GRID STRUCTURE ON THE BODY DERIVED FROM GEOMETRICAL SCHEMATIC C.

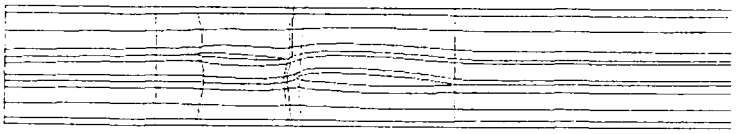


FIGURE 13. GRID STRUCTURE ON THE BODY DERIVED FROM GEOMETRICAL SCHEMATIC A.

The additional blocks of grid introduced between the canard and the wing prove necessary to resolve the region between the two surfaces in the region of the canard tip. Ideally, this region should be further resolved using a grid embedding approach or the introduction of regions of unstructured grid[9]. However, the flexibility of our approach enables a wide range of grid structures to be investigated and the automatic nature of the procedure ensures that the process can be performed quickly and efficiently.

Sections of the field grid, together with the grids on the configuration, are shown in Figure 14. As previously noted, the component adaptive nature of the grid structure is evident.

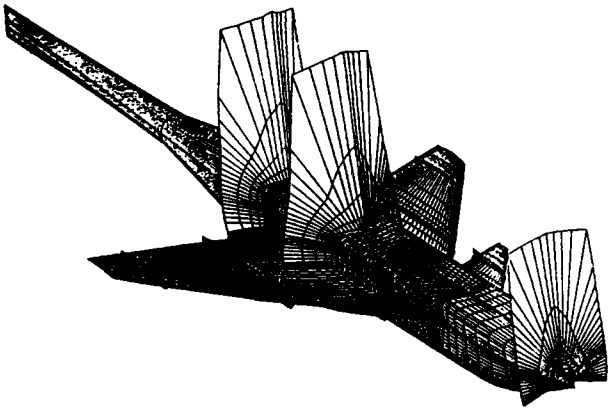


FIGURE 14. FIELD GRID SECTIONS HIGHLIGHTING THE COMPONENT ADAPTIVE TOPOLOGY STRUCTURE.

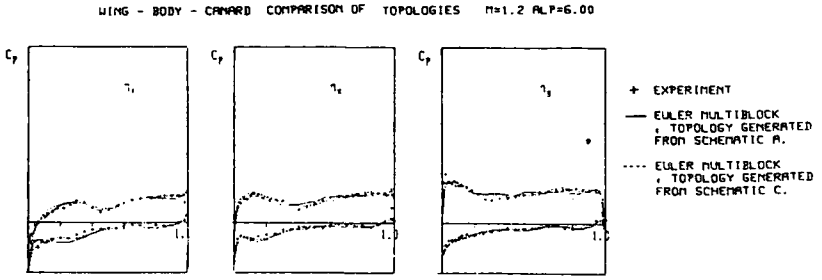
FLOW CALCULATION

In the absence of any strict criteria, the acceptability of a grid is judged firstly by eye and secondly by its performance with a flow algorithm. We have endeavoured to prove our grids by computing the flow over the wing-body-canard configurations using a numerical algorithm for the solution of the Euler equations. The algorithm, based on the ideas of Jameson, Schmidt and Turkel[10], was developed by the British Aerospace Euler Core Team at Filton, Bristol and accepts block structured grids. An example of theoretical predictions for the flow over the wing in configuration B is given in Figure 15 in which the onset Mach number was 1.2 at an incidence of 6°. For comparison the experimental data is also presented.

The good agreement with experiment for the two topologies generated by schematics A and C is evidence that, given an accurate flow solution algorithm, the grids generated from our method provide the basis for meaningful flow simulations.

CONCLUSIONS

A method has been presented which is capable of generating component adaptive grids. The approach has been illustrated using wing-body-canard geometries but is applicable to a wide range of complex aerodynamic configurations. The new method of topology generation, combined with the approach taken to grid control, provide a powerful means of exploring the most suitable topology for a given geometry. Grid control parameters are available to the user to modify the grids for particular geometries but the system does not require the user to partake in long interactive sessions on a work station to generate grids. The suitability of the component adaptive grids for flow simulation has been demonstrated by comparing theoretical predictions with experiment.



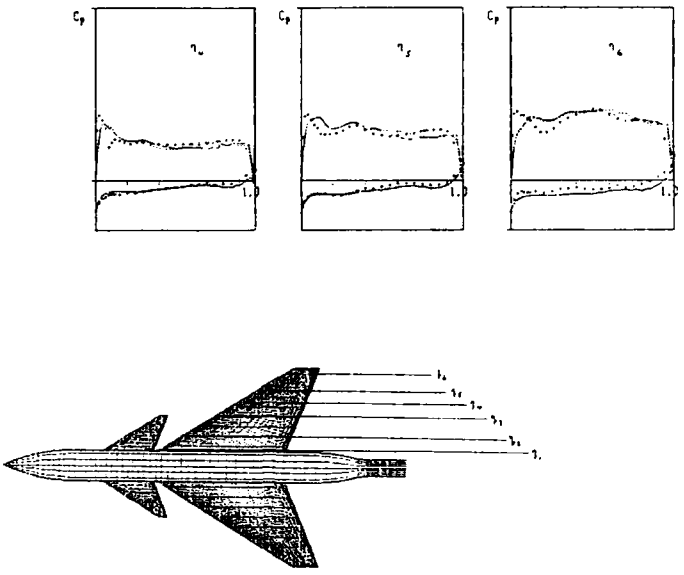


FIGURE 15. PRESSURE DISTRIBUTION ON THE WING FOR GEOMETRY A. COMPARISON BETWEEN THEORY AND EXPERIMENT.

ACKNOWLEDGEMENTS

This work has been carried out with the support of the Procurement Executive, Ministry of Defence and the Aircraft Group, British Aerospace Filton. The authors are grateful to the British Aerospace Euler Core team for use of the disc-based algorithm used to compute the flow results presented in Figure 15, to Mr. C.R. Forsey for his contributions in the early development of our multiblock work and to Dr. K.E. Rose for the development of the graphics software.

REFERENCES

1. Carr, M.P., Forsey, C.R., Developments in Coordinate Systems for Flowfield Problems. Conference on Numerical Methods in Aeronautical Fluid Dynamics, Reading, England, March 30 - April 1, 1981. Published by Academic Press. Editor P.L. Roe, 1982, pp.75-114.

2. Weatherill, N.P., Forsey, C.R., Grid Generation and Flow Calculations for Aircraft Geometries, J Aircraft, Vol.22, No.10, October 1985. pp 855-860.

3. Weatherill, N.P., Shaw, J.A., Forsey, C.R., Grid Generation and Flow Calculations for Complex Aerodynamic Shapes, Proceedings of the International Conference on Numerical Methods for Fluid Dynamics, Reading, england, April 1985. Published by Oxford University Press.

4. Weatherill, N.P., Shaw, J.A., Forsey, C.R., Rose, K.E., A Discussion on a Mesh Generation Technique Applicable to Complex Geometries, AGARD Symposium on Applications of Computational Fluid Dynamics in Aeronautics, Aix-En-Provence, France, April 1986.

5. Shaw, J.A., Forsey, C.R. Weatherill, N.P., Rose, K.E., A Block Structured Mesh Generation Technique for Aerodynamic Geometries, First International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Landshut, West Germany, July 1986. Published by Pineridge Press, Swansea,UK.

6. Thompson, J.F., Thames, F.C., Mastin, C.W. Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing any Number of Arbitrary Two-Dimensional Bodies, J.Comp.Phys., Vol.15, 1974.

7. Thomas, P.D., Middlecoff, J.F., Direct Control of Grid Point Distribution in Meshes Generated by Elliptic Equations, AIAA Journal, Vol.18, June 1980, pp 652-656.

8. Coons, S.A. Surfaces for Computer Aided Design of Space Forms, 1967 MIT MAC-TR-41.

9. Jameson, A, Baker, T.J., Weatherill, N.P., Calculation of Inviscid Transonic Flow over a Complete Aircraft, AIAA Paper 86-0103, AIAA 24th Aerospace Sciences Meeting, Reno, Nevada, January 1986.

10. Jameson, A., Schmidt, W, Turkel, E, Numerical solutions of the Euler Equations by Finite Volume Methods using Runge-Kutta Time-stepping Scheme, 1981, AIAA Paper 81-1259.

4.5

GENERATION OF MULTIPLE BLOCK GRIDS FOR ARBITRARY 3D GEOMETRIES

J.P. Steinbrenner, S.L. Karman, Jr., and J.R. Chawner
General Dynamics Fort Worth Division
Fort Worth, Texas, 76101, USA

SUMMARY

A grid generation procedure has been developed to create complex block grid systems, beginning with the generation of block surfaces, up to the generation of the full block volume grids. The multiple block concept is shown to facilitate the gridding of very complex geometries and also to allow larger sized grids to be run with a multiple block Euler solver. The entire grid generation process is broken into logical steps, each step described in detail. Three examples of grids systems generated with these techniques are given, thereby validating the procedure. Finally, current research topics in grid generation and future plans are discussed.

INTRODUCTION

One of the traditional impediments to the computational fluid dynamic analysis of complex aircraft configurations has been the inability to generate a suitable three dimensional grid efficiently. A suitable grid is defined as one which accurately describes the configuration geometry and provides sufficient resolution of flowfield phenomenon (as determined by the local truncation error of the governing differential equations) while remaining consistent with computer core memory limitations. The material presented in this paper is the result of a three year effort at General Dynamics to develop a three dimensional grid generation package applicable to arbitrary configurations. Procedures developed during this period have met these goals to a certain degree, in that an arbitrary grid system may now be generated with the codes in an efficient amount of time. However, several problems still remain unresolved or unaddressed.

This paper begins with a brief discussion of the multiple block philosophy used in the General Dynamics flow solvers. Following this is an overview of the methodology used in generation of three dimensional grids, including the generation of three dimensional surface grids, the assembly of surfaces into three dimensional volume grids, and the assembly of volume grids into multi-block grids systems. The utility of the grid generation procedure is then demonstrated through discussion of three dimensional grids surrounding three complex configurations: the F-16 fighter aircraft, a delta wing/body configuration, and an afterbody of a generic hypersonic vehicle. The nuances particular to each grid system are summarized, and any difficulties encountered in the overall grid synthesis procedure are explained. In closing, future requirements and in-work developments in grid generation are discussed.

MULTIPLE BLOCK TECHNIQUE

The underlying idea of the subject multiple block scheme is to reduce a geometrically complex region into several smaller, more manageable regions, referred to as blocks. Each block is represented mathematically by a number of discrete grid points, ordered in a three dimensional array of constant dimensions. The flowfield may be divided into any conceivable structure provided that cell to cell matching on block boundaries is maintained. This does not require that one wall of a given block match exactly with a wall of another block, only that each cell on an interface wall match with a cell of an interface wall somewhere in the grid system. The requirement of cell to cell matching was chosen to eliminate complex interpolations between blocks and to circumvent flowfield conservation problems across boundaries.

There are numerous advantages to multiple block schemes, and five of the more significant implications are summarized below.

1. The domain surrounding a complete aircraft or aircraft component is generally too geometrically intricate to model with a single three dimensional grid. This is the case, for example, with the undersurface of an F-16 fighter aircraft. The vehicle topology in the inlet diverter region is such that use of a single three dimensional grid would result in considerable skewness of grid lines. By utilizing several separate blocks in this region, however, the aircraft geometry can be accurately modeled while maintaining nearly orthogonal grid lines. The F-16 grid is described in greater detail later and also in Reference 1.

2. For increasingly many applications, a large number of grid points is needed to resolve accurately the most salient features of the particular flow field. This often creates storage requirements beyond the memory limitations of the computer. In these cases another advantage of the multiple block grid scheme becomes apparent. Since

multiple block flow solvers require that only one grid block occupy core memory at a time, while the remaining blocks reside on disk or in solid state storage, a greater total number of grid points may be used if they are divided into smaller, less memory intensive blocks. This advantage also holds for simple topologies where the grid generation is not complex but a large number of grid points is still required.

3. Because boundary interfaces need to match on the cell level only, it is possible for two adjacent blocks to have different topologies. Therefore, a combination of grid topologies may be used to model a given geometry, with each block size and topology chosen to produce the relative grid resolution needed. Proper grid resolution is also more easily controlled by using a large number of blocks by effectively increasing the number of specified grid points in the system.

4. By breaking the domain into a number of blocks, grid singularities can be placed on block boundaries and can sometimes be eliminated altogether. The branch cut for an H-grid about an airfoil, for example, becomes a block boundary if the flow field is divided into two blocks, one above the airfoil and one below. Hence, no special coding of the flow solver is necessary to handle the boundary conditions for an internal plane of the grid.

5. Normally a flow domain is divided into segments which approximately correspond to a particular aircraft component such as the nozzle, forebody, wing, or tail. This simplifies post-processing of the flow field solution. In graphical display, for instance, not all of the flow field domain must be displayed to view the flow phenomena about a given component.

Along with the advantages of multiple block systems comes an inherent disadvantage: that of the difficulty in dividing the domain into suitable blocks. Several competing considerations come into play when determining sub-domain boundaries, such as relative clustering, individual block dimensions, and physical block sizes and shapes. The fact that each block influences the remaining blocks only compounds the problem. Although work has begun in developing artificial intelligence techniques to aid in this process, currently no automated means of subdividing domains exists; the user must rely on experience, either acquired or borrowed. This continues to be a serious roadblock to the very fast general grid generation methods, and is one of the reasons why there is a steep learning curve for new users. For each of the applications to follow there was no definitive way to block the domain, and in the case of the F-16, several attempts were made before the eventual topology was determined. The multiple block technique is described in greater detail in Reference 2.

AUTOMATED GRID GENERATION METHODOLOGY

Assuming that blocking considerations, grid dimensions and the general topologies have been ascertained by some means, the grid generation process continues with the transfer of the ideas from concept to reality. Over the past three years, a series of computer programs have been developed at General Dynamics which take the grid system from beginning to end in a straightforward, logical process. The concepts built into these programs are described below.

Typically there are several individual blocks in a given system, each block having three varying computational coordinates. On each block, then, there are six faces, each face with two varying computational coordinates. Furthermore, on each face there are four edges, containing only one varying coordinate. Grid generation proceeds from the inside out, starting with the generation of face edges, followed by the determination of face surface distributions, and ending with the computation of block volume distributions. Since each step is influenced by earlier steps, it is sometimes necessary to jump backward and forward in the process, until the desired grid system is obtained. Fortunately this is easily done with the existing methods.

The first program used in the process, an interactive surface grid generator, performs two of the first three tasks. Originally written with a minimal amount of computer graphics, this program has recently been converted for use on a Silicon Graphics Iris Workstation, and has been updated considerably to take advantage of the machine's outstanding graphics capabilities. This improvement alone has cut the surface grid generation time by at least fifty percent, compared with earlier methods.

In generating a surface grid, there is usually a constrained surface on which the resulting grid must lie. This is the case, for example, with the grid used to describe the external geometry of the F-16 aircraft shown in Figure 1. An exception to this is block interface surfaces interior to a flow domain, where only a degree of smoothness is necessary, and not a specific shape. The shape of the constrained surface is often difficult to represent analytically, and so numerical models are used. Suitable models consist of a number of patches, each patch containing an M x N number of well-ordered data points. Collectively these patches are referred to as database networks, and there are only a few restrictions on their form. Databases may overlap, have different dimensions, collapse to a point, or close on themselves. Furthermore, database interfaces do not need to match exactly or to be oriented consistently. Their sole purpose is to insure that resulting grid points adhere to the surface contour of the geometry. The F-16 depicted in Figure 1 is an example of a fifty patch database network.

This network was used to generate the F-16 grid system described later.

Databases are proving to be a convenient method of surface definition, since design groups often have the ability to generate numerical models of their configurations. Work is currently underway to develop an efficient method of transferring configuration data on a CAD/CAM system to a form usable in the grid generation process, and has already met with a degree of success.

With an acceptable database of the geometry to be modelled, the surface grid begins construction. As mentioned, the first grid points to be determined are edges of the block faces. After selecting one of the six faces to work with, one of that face's four edges is chosen. The physical shape of the edge is defined by a number of arc segments chosen by the user and pieced together to form one discrete representation of the boundary shape, made continuous through the use of exponential splines (Reference 3). Arc segments may be pulled from databases, input interactively, read from exterior files, or constructed as straight line or circular arc segments. The user then divides the continuous boundary into sub-boundaries, placing a chosen number of grid points into each sub-boundary. Points are distributed by any of several techniques, the most popular of which is a two-sided stretching function developed by Vinokur (Reference 4)

When each of the four face edges have been constructed, the face interior points are given provisional values. Transfinite interpolation is used for this purpose, but with interpolant functions as suggested by Soni (Reference 5), which maintains clustering near boundaries to a higher degree than conventional interpolants. Although this technique provides a very good initial solution, it is still usually necessary to use an elliptic solver to smooth slope discontinuities and to enforce interior point clustering. The technique employed is the Thomas (Reference 6) scheme, which incorporates the standard Thompson (Reference 7) elliptic grid method in two dimensions with additional terms added to account for the curvature of the surface shape. This technique generally creates smooth distributions of grid points along the constrained surfaces, but the exact distributions are determined by the choice of weighting functions employed. Two techniques are available: the Thomas and Middlecoff (Reference 8) method; and a variation of the Sorenson (Reference 9) method, extended to three dimensional surface shapes, rather than planar surfaces.

Successive over-relaxation is used to advanced the discretized equations toward convergence, and the program allows the user to view the grid as it converges, stopping the process at any particular time. As a fully three dimensional surface is created, x and y values are calculated directly from the grid solver, and surface-conforming z-values are updated through isoparametric interpolation from the database networks. Since most grids do not lie primarily in the x-y plane, it is possible in the grid program to rotate the grid interactively to an orientation which would allow the surface shape (z) to be calculated as a function of x and y. When the grid cannot be rotated to an orientation where the surface is not double valued anywhere (more than one z value for a given x and y), as is the case for many internal flow applications, two possible remedies exist. First, the user may subdivide the surface into a number of subfaces, solving on each subface in an acceptable orientation, until the entire face is sufficiently defined. Secondly, the user may engage an alternate elliptic solver - one written in parametric rather than physical variables - whose parametric coordinates correspond to the M and N indices of the database. This technique is described in detail in Reference 10. The latter method allows the entire face to be solved at once, and is considerably faster than the physical variable solver, because the time-consuming search algorithms in the z-interpolation routine are no longer necessary. Unfortunately, the utility of this technique depends on the ability of the surface to be represented by a single database, which is sometimes difficult. However, the combination of both techniques have allowed any surface grid encountered to date to be created without significant difficulty.

There are many other features incorporated into the surface grid generation program which add to the code's speed and efficiency. For example, the latest versions allow all six walls of a given block to be generated in one interactive session, maintaining point continuity on all twelve block edges as the block is generated. This eliminates the cumbersome and confusing task of copying boundaries of one wall into a boundary of another adjacent wall and then assembling all six faces together properly. Also, as mentioned earlier, it is possible to break a face into any number of subfaces, which may either overlap, coincide or neither. By so doing a certain region of the face may be fixed in space while the other points move toward convergence, essentially allowing non-rectangular computational regions to be generated.

Experience has shown that an interactive surface grid generation scheme affords the user a very high degree of control over grid point placement, and extensive graphical capabilities add to the ease in which a grid may be constructed. Message windows added in the latest versions display diagnostic information which reduces confusion and eliminates the duplication of work. A sample screen from an IRIS Workstation during a typical grid generation session is supplied in Figure 2 indicating the layout of the diagnostics. Output from this code are files which contain all six walls of a given block, and the entire process is repeated for each of the remaining blocks in the system. The interactive grid procedure is documented in Reference 11.

Distribution of grid points on the interiors of each block grid is the final step in

the process. As in surface grids, it is necessary first to assign provisional values to interior grid points before an elliptic solver is called. Transfinite interpolation is again chosen, with interpolants calculated in the manner of Soni (Reference 5). Again, this scheme usually provides a good degree of clustering throughout the grid, but local regions of crossed grid lines, corresponding to negative values of the three dimensional Jacobian, sometimes result, particularly when there are sharp corners or very large degrees of clustering on the boundaries. When negative cell volumes exist, there are several options available, all of which utilize the three dimensional elliptic grid generation equations popularized by Thompson et al. (Reference 7). The first option is to calculate the cell volumes throughout the grid, flag the negative volumes and volumes which border the negative volumes, and to solve the elliptic equations at all of the flagged points. In this option, weighting functions are set equal to zero. This method will eliminate all of the negative Jacobians, but will not eliminate discontinuities in grid line slopes which occur from transfinite interpolation. If the discontinuities are severe, a second option is used. Here, weighting functions which correspond to the current grid point locations are calculated, with the mixed-derivative terms eliminated to allow for a greater degree of smoothness. The equations are then solved iteratively towards convergence. Still another technique is used when neither method above proves adequate. The GRAPE technique, developed by Sorenson (Reference 9), will allow for an exact grid point spacing and transverse angle specification at the boundaries of the six walls of the block. This method is particularly attractive when strict orthogonality is desired at one or several of the block walls, or when very tight clustering is needed at the walls. All of the three dimensional techniques described employ an successive over-relaxation scheme as a means of advancing the numerical solution. An Approximate Factorization scheme is also available, but has not yet been found to be superior to the point relaxation scheme.

None of the three dimensional schemes described above will produce acceptable interior grid point distributions for every conceivable set of block walls. In fact, it is doubtful that such a scheme will exist in the near future. For this reason, a greater emphasis is placed on careful generation of surface grids which will not force the interior grid lines to follow unreasonable paths. This is possible with the surface grid generation program, which allows boundaries or walls to be created and recreated quickly.

The block grid generation process is repeated for each individual block in the system, each block generated independently. Consequently, the resulting block system generally exhibits slope discontinuities across block boundaries. The discontinuous lines can be controlled to some degree by judicious grid point distributions on adjacent faces, but it has been observed that slight discontinuities in slope present no major problems, particularly when using a finite-volume flow solver. Despite the efforts to develop an all-encompassing grid generation package, certain problems still exist, and are discussed later. Future research and development topics to further aid in the grid process are discussed later as well.

APPLICATIONS

The three examples in this section, presented in chronological order of generation, illustrate the class of configurations that can now be treated on a fairly routine basis. These examples were created as the grid generation programs evolved, and in fact influenced the structure of the programs as new problems were uncovered.

F-16 Fighter Aircraft Grid

A three dimensional grid was generated for an Euler analysis of the F-16 fighter aircraft. The grid, which models the left half of the aircraft, contains twenty blocks with a total of 530 000 grid points. All components of the vehicle are simulated including the wing, body, horizontal and vertical tails, inlet, nozzle and ventral fins. The wing tip missile and missile launcher, however, are not simulated. The database used to define the F-16 surface geometry is displayed in Figure 1. A detailed discussion of the grid generation and Euler analysis of this configuration can be found in Reference 1.

The first step in generating this grid was development of the blocking structure. The geometry was easily divided into upper and lower domains, with the wing and horizontal tail residing in the interface plane between the two domains. In order to maintain cell to cell matching across this horizontal block boundary the grid topology for both domains needed to be the same. This became a considerable restriction in developing acceptable block arrangements for both domains.

The H-grid topology of the lower domain was selected based on the blocking requirements of the inlet diverter and ventral fin regions. Just aft of the main inlet face the geometry was simulated as shown in Figure 3. The diverter section above the inlet was discretized with one very small block which collapsed into the environmental control system inlet. The H-grid which ran alongside the inlet continued down the fuselage to the ventral fin area where the fin was aligned with the block boundary. The entire lower domain, shown in Figure 4, contains thirteen blocks. Because of the complexities of the geometry, it could not be combined into one contiguous block.

The upper domain, however, was generated in one contiguous block and then divided

into the seven blocks outlined in Figure 5. Generation of the upper domain H-grid was simplified relative to the lower domain because there were no fins or inlets in this region. The sizes of the blocks were chosen to optimize the core memory usage. A view of the upper surface grids is shown in Figure 6.

When this analysis was begun, the surface grid programs were still under development. At that time, the code required that each face of a given block be made individually, and that shared boundaries be written to a file and read in for all adjoining faces. Also, the program had no substantial graphics capabilities. At the same time, the corresponding block Euler flow solver was under development. These problems combined with the geometric intricacies caused the grid to be generated at a very slow pace. Consequently, nearly a full-man year was needed to generate the complete twenty block grid system.

Delta Wing/Body Grid

Later, a three dimensional grid was generated for an Euler analysis of a supersonic delta wing configuration. This configuration, shown in Figure 7 was obtained from References 12 and 13. The geometry included a thin delta wing, slender body and a vertical tail. Lateral and longitudinal stability derivatives were to be computed from the Euler flow field solution. Since there was no left-right symmetry in the flow field, the entire aircraft was modelled. The complete multiple block system contains four blocks and 200 000 points. This configuration is geometrically simpler than the F-16 aircraft, and at the time of generation, the grid generation codes were more advanced and the block Euler code was fully operational.

The block arrangement was also simple compared to the F-16 due to the lack of propulsion system components and auxiliary control surfaces. The four blocks are arranged to discretize the four quadrants about the aircraft. The wings and the vertical tail are simulated as part of the interface planes between the upper and lower blocks and the symmetry plane is the natural boundary between the left and right blocks. The farfield boundaries were positioned close to the body because the Euler analysis was to be done at supersonic conditions. The outer boundary is extended upstream of the nose just far enough to capture the bow shock and the downstream face of the grids is positioned at the end of the body and vertical tail.

Figure 8 shows the boundaries of the block of grid used to describe one of the upper quadrants. A location on the body at midwing is arbitrarily selected as a face boundary. The grid in the cross planes is then established as an O-grid while the grid in the transverse planes is a C-grid. This type of topology results in a singular line of grid points extending forward of the nose. The complete grid on five of the six faces of this block is shown in Figure 9. Generation of the lower quadrant block proceeded in a similar manner. Then, the symmetrically opposite blocks were generated by reflection of the two existing blocks.

Less than a man-week was needed to generate this block system, and there are several reasons for this significant reduction in manpower. The foremost region was the large amount of experience gained from generation of the F-16 grid which was directly applicable to this geometry. Also, an advanced version of the grid generation procedure was available which employed extensive interactive graphics on an IRIS Workstation. The biggest single reduction in manpower was due to the simplicity of the geometry, but the program enhancements helped considerably as well, probably speeding the entire process up by an order of magnitude.

Hypersonic Vehicle Afterbody Grid

A three dimensional grid was generated for Euler analysis of the afterbody and nozzle region of a generic hypersonic vehicle. The geometry of this symmetric region is defined by the database shown in Figure 10. The afterbody has a rounded cross section which necks down to a sharp trailing edge. The underside of the expansion ramp is aligned at twenty degrees with respect to the horizontal. The engine module on the underside of the afterbody has been approximated by a thin walled rectangle with sidewalls that extend approximately one half the length of the lower flap. The outflow boundary of the grid is located one afterbody length downstream of the ramp end and the farfield boundaries are conically shaped.

The first step in generating this grid was to develop the blocking structure. The blocking arrangement that resulted was based on three considerations. The first issue dealt with was the differing shapes of the inside of the engine module and the exterior of the afterbody. The rounded afterbody shape made a C-grid in this region most advantageous. However, in order to avoid a line singularity at the center of the engine flowfield, an H-grid was chosen over a C-grid for the internal region.

Having defined the blocking based on grid topologies the next issue considered was the block matching in the circumferential direction. The connection of a C-grid to an H-grid as shown in Figure 11 would require that the grid points used on the perimeter of the H-grid match with the inner radial boundary of the C-grid. The engine module presented a particular challenge here. Due to the requirement of point to point matching between blocks, block boundaries were set on the corners of the engine module and

extended radially outward to the farfield boundary. These block boundaries required that the number of points on the perimeter of the internal H-grid match the number of points on the inner radial boundary of the C-grid. Based on these considerations the external grid was blocked into three circumferential sections corresponding to the top, side, and bottom of the engine module.

The final issue considered in blocking this afterbody grid was one of number of grid points per block. At this point the grid had been divided into a minimum of four blocks: a rectangular H-grid surrounded by three sections of C-grid. The total number of grid points was limited to 400 000 in order to keep flow solver run times at an acceptable level. Considerations of geometric accuracy led to the decision to use 126 points in the streamwise direction (96 points up to the end of the afterbody), 81 points in the circumferential direction, and 21 points in the radial direction. In order to satisfy computer core memory limits the number of points per block needed to be approximately 30 000. This was easily accomplished by adding block boundaries at two streamwise stations, namely at the end of the lower flap (46 points) and at the end of the upper flap (51 points).

The grid for this afterbody has been divided into twelve blocks as shown in Figure 12; one radial block boundary separates the H- and C-grids, two circumferential block boundaries define the shape of the engine module, and two streamwise block boundaries enforce the limit of grid points per block. The blocking process described above was not very difficult but still consumed approximately two days since it was all done by hand. Much of the delay was a result of selecting the proper number of grid points per block. The blocking decisions were simplified by the smooth lines of the afterbody geometry. A graphical procedure on a workstation employing some form of artificial intelligence could have cut the time for this task considerably.

The next step in creating this three dimensional grid was generation of the two dimensional surface grids for each block. In order to assure grid line slope continuity across block boundaries the entire external C-grid up to the trailing edge of the afterbody (three of the twelve blocks) was generated as one block. At the time of this analysis a new feature had been added to the interactive grid generation procedure which allowed for all six faces of a block to be generated simultaneously. This feature proved to be extremely helpful in assuring point continuity on the face edges. The topology for the upstream outer block is shown in Figure 13 along with the associated grid indices. Specifically, generation of the upstream face will be described since it was complicated by the concentration of points in the engine sidewall region due to the internal H-grid.

The boundaries of the upstream face are shown in Figure 14. The farfield and symmetry boundary shapes were selected based on consideration of farfield boundary condition influence on the body whereas the engine module and afterbody shapes were obtained from the database shown in Figure 10. To illustrate the evolution of a particular wall, a close-up of the circled region in Figure 14 is presented in Figure 15a, with only the boundary points displayed. The grid on this face was initialized using an algebraic transfinite interpolation scheme yielding the grid shown in Figure 15b. Obviously, the grid line crossing, lack of resolution of the corner, non-orthogonality at the corner, and small cell sizes are unacceptable. The reason for the problems in this case are the highly stretched and compressed boundary point distribution (multiple length scales) and the discontinuities in the boundary shape. Since boundary orthogonality and clustering in the corner were deemed necessary, the Sorenson weighting functions were used to solve the grid equations, and resulted in the grid shown in Figure 15c. Resolution of the corner and orthogonality has been obtained but severe pinching of the grid lines has appeared in the corner. This pinching was relieved by running the grid solver using the Thomas and Middlecoff weighting function resulting in the final grid shown in Figure 15d. Once this upstream face was completed work on the remaining five faces of this block continued. This process required several iterations with the grid solver, changing the weighting functions from one formulation to the next in order to obtain the described grid point distribution. The interactive graphics employed in the grid generation program allowed the user to view the grid as the solver progressed at each iteration. This avoided the continuation of bad solutions or gave the opportunity of stopping the solution when a good solution was obtained.

As each face of the three dimensional grid was completed its boundaries were written to the corresponding boundaries of the connecting faces, simplifying the generation of these grids. Eventually, then, the last face worked upon already had all four of its boundaries defined by the completion of the adjacent faces. A view of the four of the six faces of this complete grid block is shown in Figure 16. Generation of the remaining blocks of the twelve block grid proceeded in a similar manner.

CONCLUSIONS

The viability of the present methods to generate multiple block grid systems has been verified for only three geometries in this paper, but has been used to generate numerous other systems as well. Viability, of course, can only be ascertained after the grid is used in a flow solver, but with the techniques in this work, a reasonable degree of confidence can be obtained before the grid is taken to task. An additional program has been developed as a post-processing tool in grid generation to further increase the grid confidence. This program allows the operator to load in and scroll through a grid

block, one computational plane at a time, adjusting the scrolling speed as desired. This feature makes it possible to digest a large amount of information about the grid in a short amount of time, without cluttering the screen with unneeded information. The general continuity of the grid lines can also be determined with this program in a short period of time.

Although a grid system could be generated for almost any imaginable geometry with the present scheme, certain geometries still present a great amount of difficulty. Reasons behind the difficulties are stated below. First, the technique of breaking a domain into blocks is not easily taught, and becomes a formidable task for complicated domains regardless of the user's experience. Secondly, the total time needed to generate a grid system has been reduced considerably over the past few years, but is not yet to the level that makes a very fast (2-3 days) analysis of a configuration possible. Part of this particular problem is due to the speeds of the computers used, but there is also a need for further automation of the grid generation process. Finally, when viewing a three dimensional grid, it is difficult to determine if everything is as it is expected to be, especially with grids of large dimensions. This becomes even more difficult with multiple block systems, where grid boundaries connect to others in a pre-specified manner.

These problems have been the primary impetus for continued development of grid generation methods at General Dynamics. As mentioned earlier, artificial intelligence techniques have begun to be used to assist in the domain blocking procedure. These methods may not yet be ready for use in three dimensional block grids, but a set of rules in determining blocking boundaries are finally beginning to be formulated. Currently, after determining block boundaries, a file is normally created by the user which defines the connectivity of the block grid system. This file is then input to the flow solver, and the grid is generated independently. Work is already underway in developing a program to allow the connectivity file to be created in a interactive graphical environment. Conceptually, this program would output two files - one each for the flow solver and the surface grid generator. The corresponding connectivity file would be attached to the grid generator, and the entire block system would be generated, with connectivity automatically maintained as specified by the file. This alone would reduce the throughput time for grid generation considerably. By having a connection file, it would also be relatively easy to use three dimensional elliptic grid schemes in the entire domain, with continuity across block boundaries. Block continuity is a problem that has been addressed and solved by some other researchers, but has not yet been implemented into the present scheme. Finally, as the size and complexity of three dimensional grids increases, so will the reliance on graphical techniques to check and validate resulting grids. Consequently, work has begun in developing improved graphical programs to view grids. It is probably not unreasonable to expect implementation of the ideas above to result in another order of magnitude reduction in the time needed to create a complex block grid system.

REFERENCES

1. Karman, S.L., Jr., Steinbrenner, J.P., and Kisielewski, K.M., "Analysis of the F-16 Flow Field by a Block Grid Euler Approach", AGARD-CP-412, 1986.

2. Reed, C.L., and Karman, S.L., "Multiple Block Grid Method Applied to Complex 3-D Geometries", presented at the Society for Industrial and Applied Mathematics 1986 National Meeting, Boston, MA.

3. McCartin, B.J., "Theory, Computation, and Application of Exponential Splines", DOE/ER/03077-171, October, 1981.

4. Vinokur, M., "On One-Dimensional Stretching Functions for Finite-Difference Calculations", NASA CR-3313, October 1980.

5. Soni, B.K., "Two- and Three-Dimensional Grid Generation For Internal Flow Applications of Computational Fluid Dynamics", AIAA 85-1526, 1985.

6. Thomas, P.D., "Composite Three-Dimensional Grids Generated by Elliptic Equations", AIAA Journal, Vol. 20, 1982, pp. 1195-1202.

7. Thompson, J.F., Thames, F.C., and Mastin, C.M., "Automatic Numerical Grid Generation of Body Fitted curvilinear Coordinate Systems for Fields Containing any Number of Arbitrary Two-Dimensional Bodies", Journal of Computational Physics, Vol. 16, 1979.

8. Thomas, P.D., and Middlecoff, J.F., "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations", AIAA Journal, Vol. 18, 1979, pp 652-656.

9. Sorenson, R.L., "A Computer Program to Generate Two-Dimensional Grids about Airfoils and Other Shapes by the Use of Poisson's Equations", NASA Ames Research Center, NASA TM-81198, 1980.

10. Thompson, J.F., Warsi, Z.U.A., and Mastin, C.W., Numerical Grid Generation, North Holland Publishers, New York, 1985.

11. Steinbrenner, J.P., "GRIDGEN2D, Interactive Elliptic Surface Grid Generation", General Dynamics Report No. CFD 063-4-8601, June 13, 1986.

12. Jernell, L.S. "Longitudinal Aerodynamic Characteristics of Wing-Body Models With Arrow, Delta, and Diamond Planforms at Mach Numbers from 2.30 to 4.63 and Comparisons with Theory", NASA TM X-1522, March 1968.

13. Jernell, L.S. "Stability Characteristics at Mach Numbers from 2.30 to 4.63 of Wing-Body-Tail Models Having Wings With Arrow, Delta, and Diamond Planforms", NASA TM X-1485, December 1967.

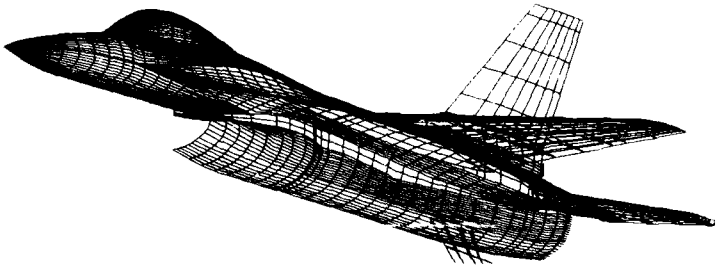


Figure 1. F-16 Fighter Fifty Database Network

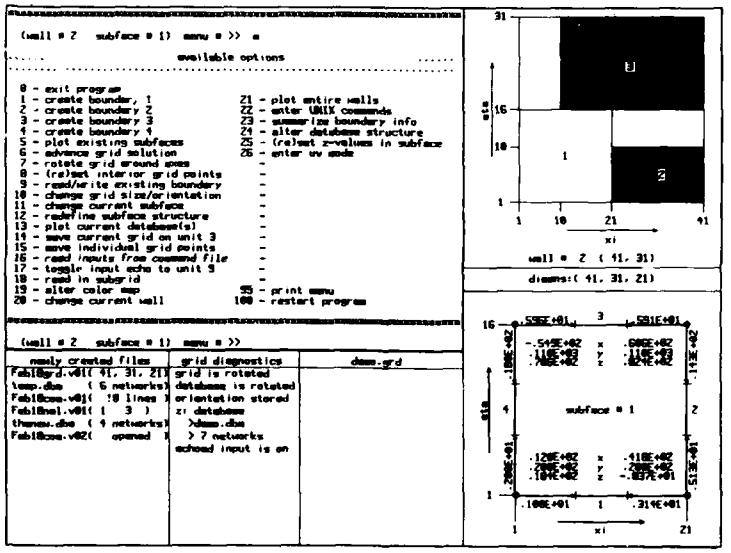


Figure 2. Sample Screen from the Interactive Grid Generation Procedure

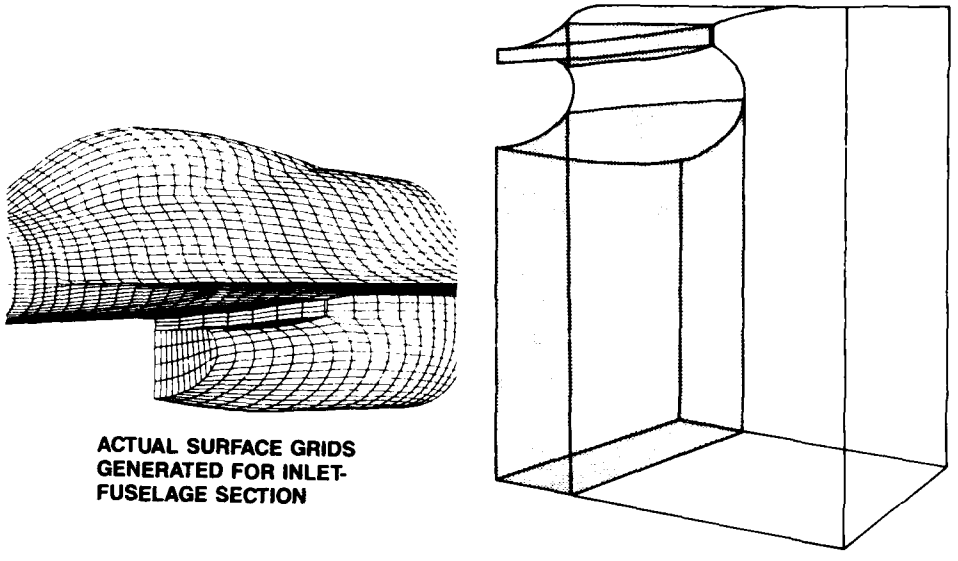


Figure 3. F-16 Inlet Diverter region Grid and Multiple Block Structure

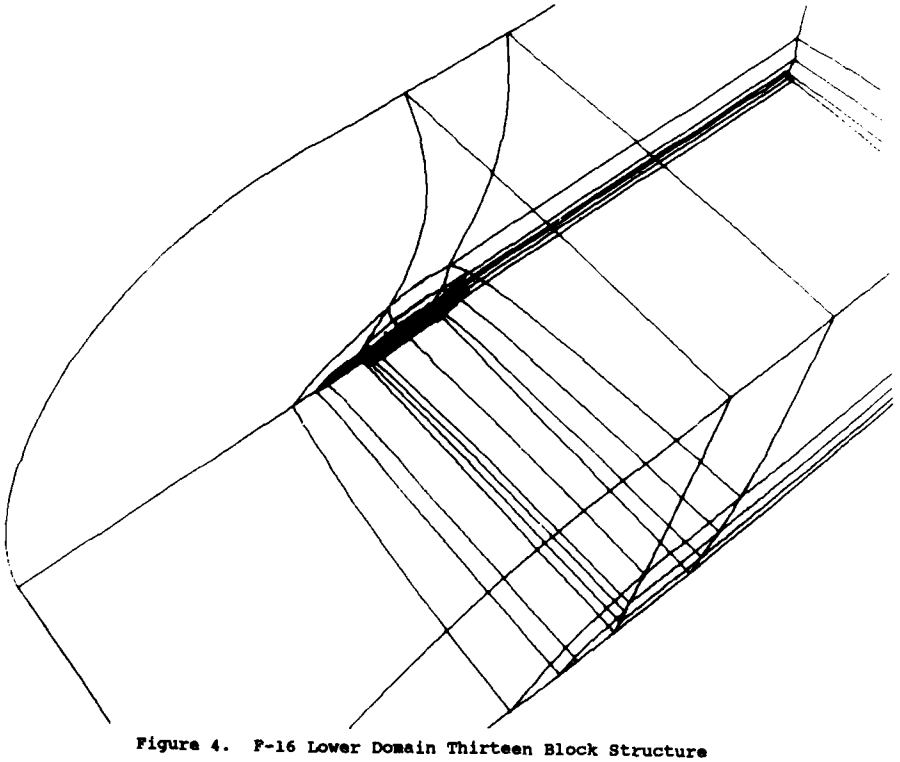


Figure 4. F-16 Lower Domain Thirteen Block Structure

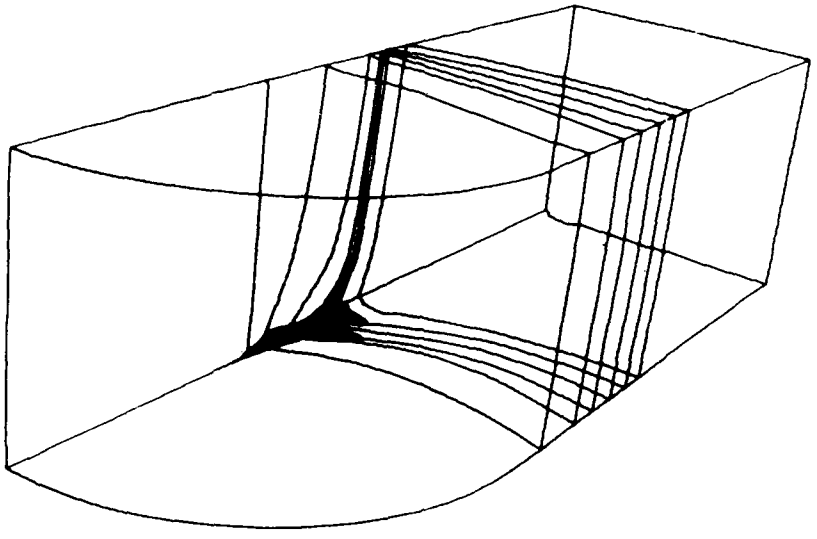


Figure 5. F-16 Upper Domain Seven Block Structure

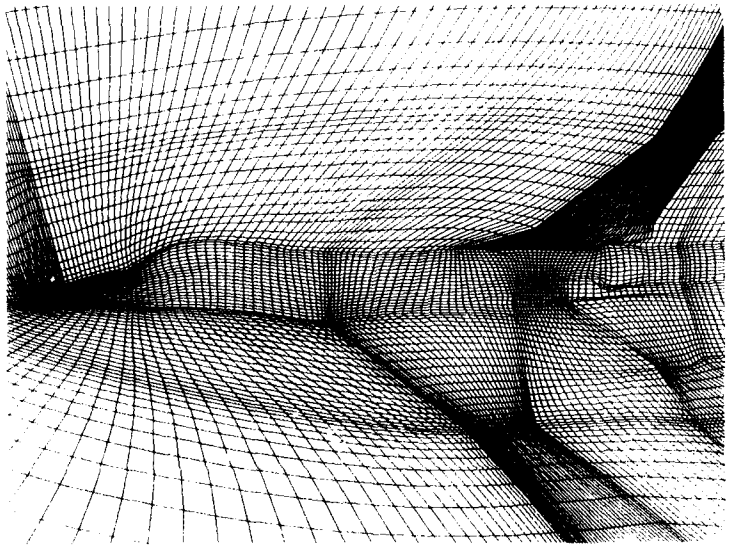


Figure 6. F-16 Upper Surface Grid

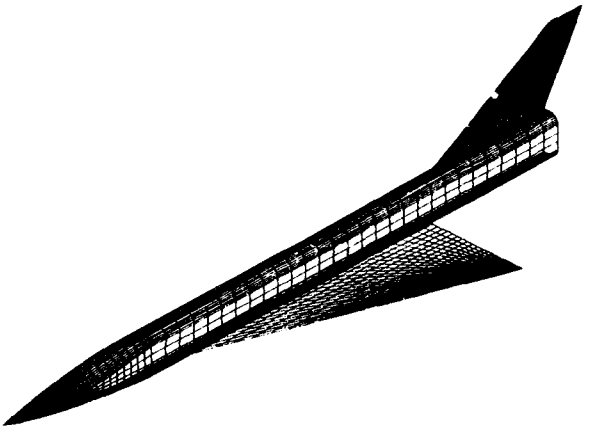


Figure 7. Delta Wing/Body Four Network Database

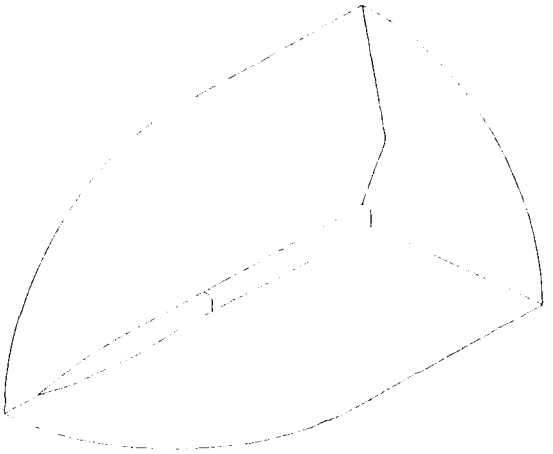


Figure 8. Delta Wing/Body Upper Right Quadrant Block Face Boundaries

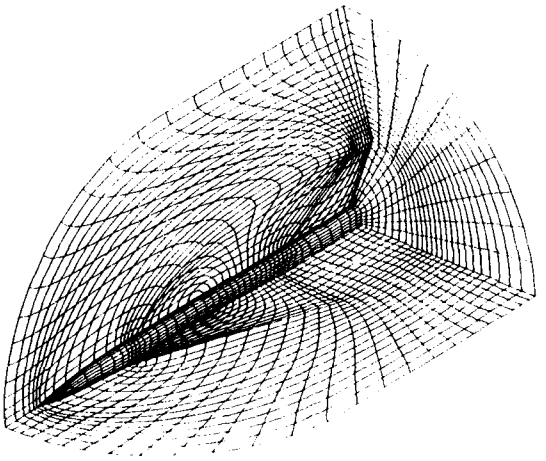


Figure 9. Delta Wing/Body Upper Right Quadrant Grid (reduced)

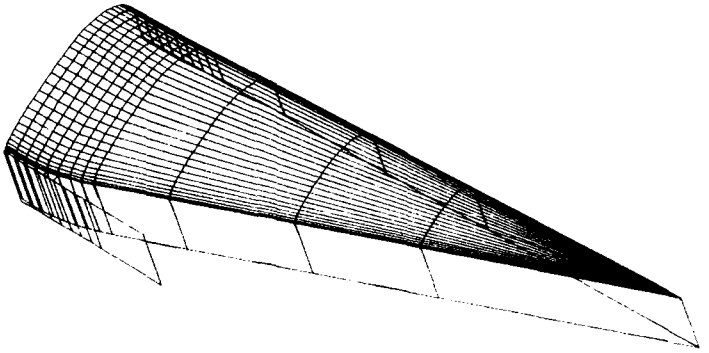


Figure 10. Generic Hypersonic Afterbody/Nozzle Three Network Database

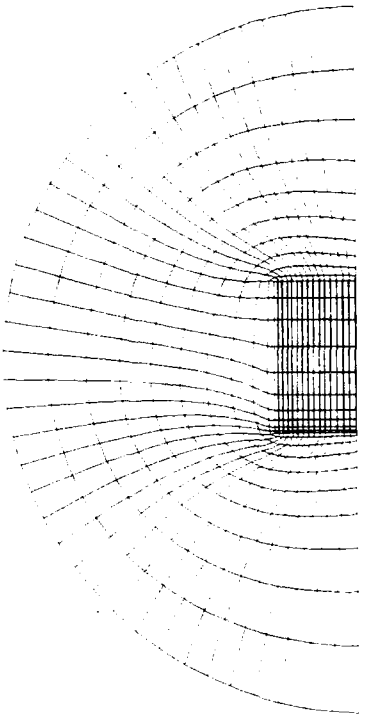


Figure 11. Generic Hypersonic Afterbody/Nozzle H-Grid to C-Grid Interface

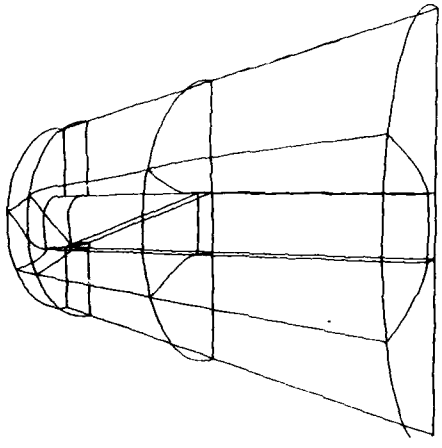


Figure 12. Generic Hypersonic Afterbody/Nozzle Twelve Block Structure

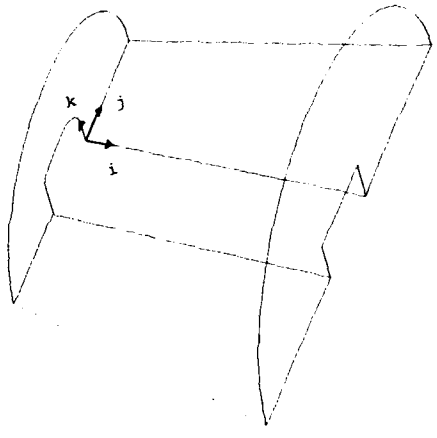


Figure 13. Generic Hypersonic Afterbody/Nozzle Upstream External Grid Block Boundaries

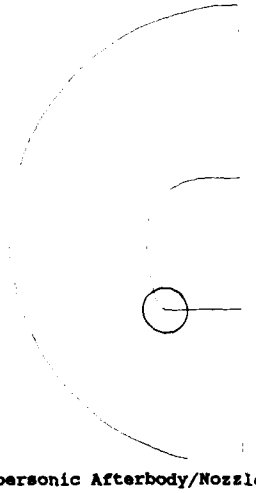


Figure 14. Generic Hypersonic Afterbody/Nozzle Upstream Face Boundaries

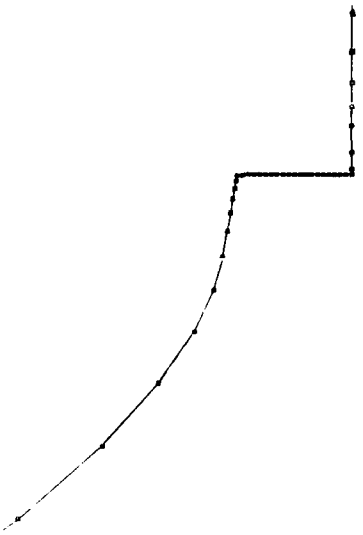
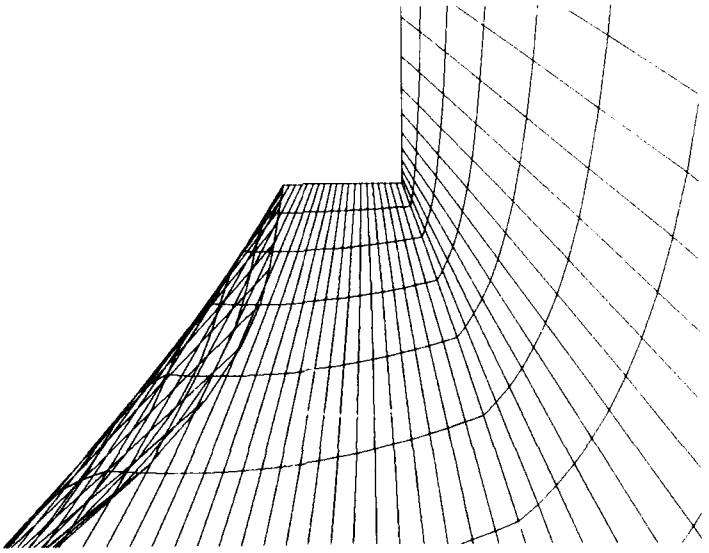
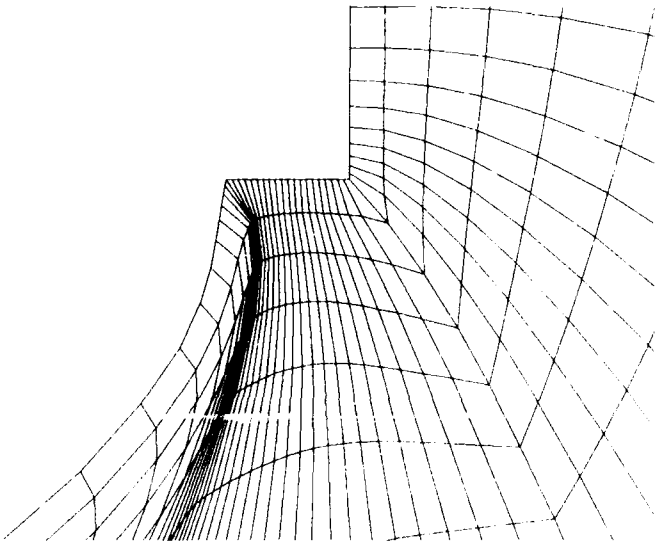


Figure 15. Generic Hypersonic Afterbody/Nozzle at Engine Module Sidewall Region

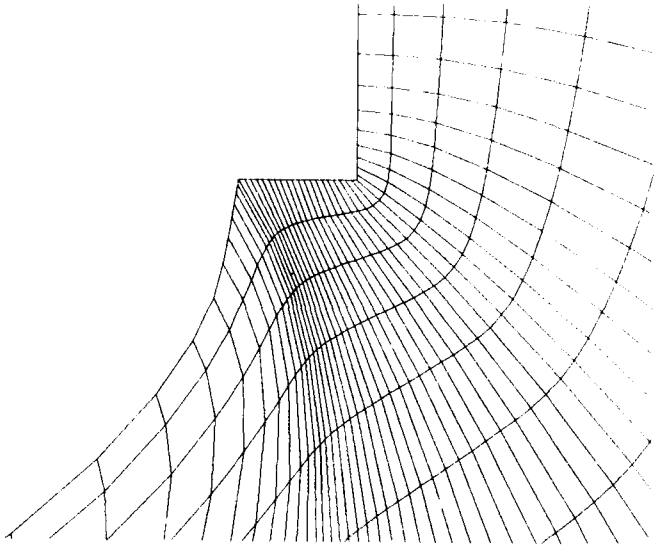
a. Boundary Point Distribution



b. Transfinite Interpolation Grid



c. Sorenson-type Weighting Function Grid



d. Thomas and Middlecoff Weighting Function Grid

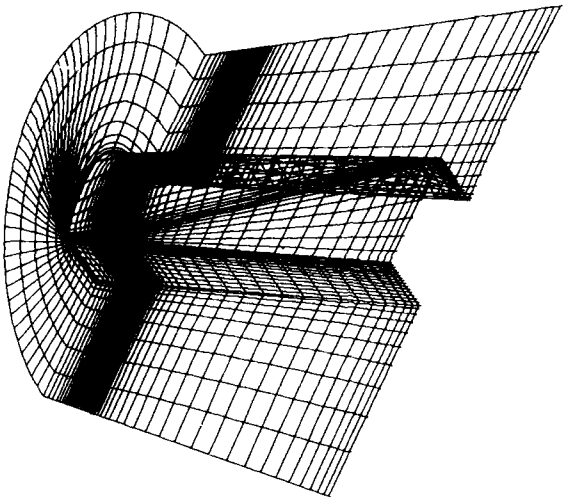


Figure 16. Generic Hypersonic Afterbody/Nozzle External Grid Block Boundary Point Distribution

4.6 GRID GENERATION ON AND ABOUT A CRANKED-WING FIGHTER AIRCRAFT CONFIGURATION

by

Robert E. Smith and Joan I. Pitts
NASA Langley Research Center
Hampton, Virginia 23665

Lars-Erik Eriksson
Old Dominion University
Norfolk, Virginia 23508

Michael R. Wiese
Computer Sciences Corporation
Hampton, Virginia 23666

SUMMARY

This paper describes experiences at the NASA Langley Research Center generating grids about a cranked-wing fighter aircraft configuration. A single-block planar grid about the fuselage and canard and used with a finite-difference Navier-Stokes solver is described. A dual-block nonplanar grid about the complete configuration and used with a finite-volume Euler solver is described. The very important aspect of computing the aircraft surface grid--starting with a standardized model description--is also described.

1. INTRODUCTION

In 1984 an effort was initiated at the NASA Langley Research Center to compute grids and flow fields about a complex fighter-type aircraft configuration consisting of the following four components:

1. an area-ruled fuselage with a canopy and integrated engine inlet,
2. a bi-convex swept canard,
3. a bi-convex 70-20 cranked wing,
4. a swept vertical tail.

An orthographic view of the four components is shown in Figure 1, where each component is described by an ordered set of cross sections relative to a primary axis.

The initial step for flow field computation about a configuration such as the fighter model described above is to establish the number, placement and topology of grid blocks that cover the physical domain. In our experience the best way to approach the problem is to sketch the configuration and outlines of the physical blocks. In deciding on the block structure and topology, it is important to consider the type of flow calculations that are to be made and the characteristics of the solution algorithm. The grid itself with its concentrations and dispersions is not of great concern at this early stage, but the connecting of blocks and the singularities in the blocks are important.

The second step is the detailed computation of the grid on the configuration surface subject to the chosen topology. This implies that grid points will likely have different locations from the defining cross section points. Also, grid curves on the configuration may have a different orientation from the defining cross sections. We have used a bi-cubic representation (Coons' patches) and corresponding software that has been developed at the Langley Research Center to mathematically represent aerospacecraft (Refs. 1 to 3). The intersection of components and the grid curves on the component surfaces are computed from patch-plane intersections where the planes are user prescribed. The patch-plane intersection capability is also a part of the Langley surface definition software. In order to use the surface definition software for grid generation, it is necessary to write a driver code to call the surface generation code, and to manipulate and manage data for the chosen topology.

Once the configuration surface grid is determined, the next step is the generation of the surrounding grid for each block. Our general approach is to work from the configuration surface out to the exterior far field boundaries. The remaining steps are the flow field computations and their analysis. A point that must be considered here is that logic for the grid generation must be incorporated into the flow field solver and subsequent visualization or analysis programs.

Two grid topologies are described herein. The first grid has a single block (Figure 2) with no singularities. This grid extends from a point just behind the nose of the configuration back to where the the engine inlet begins and is used with a finite-difference technique to compute viscous supersonic flow. The second grid (Figure 3) is a dual-block grid above and below the canard and wing. This grid topology which is the conception of the third author has a polar singularity at the nose of the configuration and a line singularity (Ref. 4) around the fuselage at the beginning of the engine inlet. This grid is used with a finite-volume technique to compute inviscid

supersonic flow about the complete configuration with flow into the inlet. Individual discussions are devoted to our experiences with each grid structure. The surface grid generation for the fighter configuration is similar for both grids, and a description of our experiences is presented.

2. SINGLE-BLOCK PLANAR GRID

A single-block planar grid is constructed. The grid is used with a finite-difference Navier-Stokes solver (Ref. 5) to compute viscous supersonic flow over the forward part of the fighter configuration (fuselage and canard, Figure 2). The Navier-Stokes solver integrates governing equations in a computational coordinate system related through the Jacobian transformation to the physical domain (Ref. 6). The solution procedure is a time-split MacCormack technique, and no singularities are tolerated since the Jacobian derivatives appear explicitly in the equations of motion. For this reason, planar grid surfaces are started downstream of the nose of the fuselage and continue past the canard.

The computation of grid points on the fuselage and canard are discussed under boundary grid generation elsewhere in this paper. The outer boundary in each plane is a semicircle with increasing radius in the downstream direction (conical surface), and each plane is divided into an upper and lower part. The two-boundary technique (Ref. 7) with clustering distributions is applied to the upper and lower portions of each plane. The clustering is designed to concentrate grid points near the fuselage and canard surfaces and toward the canard-fuselage intersection. For the application of the two-boundary technique, the inner boundary is at the fuselage and the outer boundary is the circular segment. The left boundary is the symmetry line and the right boundary is the surface containing the canard. The left and right boundaries are reversed for the lower grid. Figure 2 shows the grid topology and selected grid surfaces. Figure 4 shows a solution of the pressure on selected grid surfaces at Mach Number = 2.5, Reynolds Number = 65,000/Meter, and 0 degree angle of attack using 278,800 grid points.

This grid topology and solution procedure were our first attempts to solve supersonic flow over the fighter configuration. There were many lessons learned. The first lesson is that the grid should not be planar. Grid lines should be aligned with the leading and trailing edges of the lifting surfaces. We believe that such grid alignment improves accuracy in the application of boundary conditions, relaxes the requirement for artificial damping, and decreases the coding complexity. A second lesson is that solving the Navier-Stokes equations with a large number of grid points (200,000 and more) is extremely taxing on the present generation super computers. At the present time there is much more potential for routinely solving the Euler equations about complex three-dimensional geometries as is demonstrated in the next section.

3. DUAL-BLOCK GRID

For the fighter configuration shown in Figure 1 a multiple-block C1-continuous grid is constructed for inviscid compressible flow computations (Euler equations) using a finite-volume technique (Refs. 8 and 9). An initial requirement is for the grid to conform to the canard and wing edges. A single-grid topology would result in a highly-skewed grid (Figure 5), and it would be difficult to concentrate grid points at the apex region of the wing where vortex flow is generated. A dual-block grid topology (Figure 6), having an inner grid which covers part of the wing and fuselage, has a singularity grid curve on the fuselage and a bounding-block grid curve along the leading edge of the highly swept part of the wing. This topology is considered to be optimal for the flow field conditions and is suitable for the finite-volume technique. Once the topology has been chosen, the next step is the computation of the grid on the configuration surface which is discussed elsewhere in this paper. However, at this point, distributions for grid clustering on the configuration surface must be established. For the fighter configuration there is clustering near the intersections of the lifting surfaces and the fuselage and the leading and trailing edges of the wing and canard. Also, there is a clustering on the wing surface from the wing crank to the trailing edge of the wing. Figure 7 shows the topology of the dual-block grid relative to the computational domain, and Figure 8 shows the surface grid in an exploded view.

The exterior grid generation about the fighter configuration is based entirely on transfinite interpolation and is therefore computationally efficient. Transfinite interpolation is the Boolean sum of several univariate interpolations in which distribution functions can be embedded for grid clustering (Ref. 10). Usually the interpolations are low order polynomial functions such as hermite cubic functions (Ref. 4). The process is to work from the configuration surface outward, computing subgrids and "gluing" them together with C1 continuity. For each subgrid, some of the six bounding surfaces are obtained from the configuration surface grid, some are obtained from previously computed adjoining subgrids, and some are constructed from simple analytic functions. C1 continuity is maintained by using derivative information as well as grid point locations, and Figure 3 shows selected grid surfaces. For the flow field computations that have been made thus far, there have been 264,000 grid points. A typical solution, showing the coefficient of pressure on selected surfaces at Mach number 2 and an angle of attack of 4 degrees, is shown in Figure 9.

4. AIRCRAFT SURFACE GRID GENERATION

The fighter configuration shown in Figure 1 has a complex fuselage with sharp corners, a swept engine inlet and a deep cavity above the engine inlet (boundary-layer diverter region). The canard and tail geometries are simple, and except for the 70-20 degree crank, the wing geometry is simple. An added complexity is that the root leading edge of the wing is near the center of the vertical side of the fuselage, and the root trailing edge is near the top. For the single block grid (Figure 2) the most complex part of the configuration is not considered. For the dual-block grid (Figure 3), the boundary-layer diverter region is omitted which requires the reconstruction of new defining cross sections in the engine inlet region. It is anticipated that the boundary-layer diverter region will be re-introduced in the future as an additional grid block.

Each component (fuselage, canard, wing and vertical tail) is mathematically represented by ordered sets of Coons' patches as described in references 1 and 2. A Coons' patch is a bi-cubic function with two parametric independent variables. The defining parameters for a patch are: (1) coordinate positions of corner points; (2) derivatives of coordinates with respect to the parametric variables at the corners; and (3) cross derivatives of the coordinates with respect to the parametric variables at the corners. There is a maximum of forty-eight parameters for each patch. The input to the Langley surface definition software are coordinates along cross sections at stations along each component. Corresponding points on neighboring cross sections become the corners of patches, and it is advisable that the distribution of input points be approximately the same along each cross section. Cubic splines are fitted along the cross sections through the defining points and across the cross sections through the defining points. Also, the parametric variables are defined along and across the cross sections respectively, and derivatives with respect to the parametric variables are evaluated at the defining points. At the present time the cross derivatives are set equal to zero in the Langley software. Coordinate positions interior to a patch are computed by evaluating along the parametric variables and Figure 10 shows an orthographic view of the fighter configuration which has been densely interpolated and presented as a solid. Grid points are computed along curves which are the intersections of planes and the patch definitions. A grid curve can cross several components if the grid topology requires it, and the parameters defining a plane (coordinates of three points) are software input. The intersection of components (fuselage-canard, fuselage-wing, and fuselage-tail) are computed using the plane-patch intersection software, where the planes are made perpendicular to the x-axis. A search approach is used to find the beginning and end of the intersections. Figure 11 shows the fuselage grid and intersections for the canard and wing, and Figure 8 shows the entire surface grid from an exploded view.

Following are two important lessons that were learned during the modeling of the fighter configuration.

1. The initial input description was too sparse for the complex detail of the fuselage. More cross sections, particularly in the canopy region and engine inlet region, were required. Also, more defining points per cross section than initially thought were required.

2. It was necessary to break the fuselage and wing into three sub-components and two sub-components, respectively. The fuselage was divided at the cross section where the canopy starts to appear and the cross section where the engine inlet begins to appear. The wing was divided at the crank. These divisions were made because of the first derivative discontinuities on the surfaces that are not acceptable in a cubic spline computation.

The decision to make these changes was reached by observing plots and images that showed bulging where it should not be and smearing where there should be a sharp change in curvature. It should be noted that using a high order model representation such as the Coons' patch description is an effort to minimize the amount of information that must be user provided and discretely stored. This is constrained, however, by the complexity of the model and the level of detail that is required. Also, it should be noted that computer graphics is an essential tool for evaluating the grid generation on the configuration surface.

The grid topology requirements on the fuselage and canard are similar for both the single-block grid and the dual-block grid. Grid curves on the fuselage are in planes parallel to the defining cross sections. For the single block grid topology, grid curves across the canard are in parallel planes and are oblique to the canard leading and trailing edges. For the dual-block grid topology, grid curves conform to the leading and trailing edges of the canard and the grid surfaces are not planar. This is also true for the vertical tail.

Grid curves are obtained using the plane-patch intersection capability of the Langley surface definition software. A grid curve consists of a set of points collected from the entire array of patch intersections with a plane. Duplicate points are removed, the points are ordered and their approximate arc lengths along the grid curve are computed. Note that the number of points to represent a grid curve is user controlled and usually should be greater than the number of grid points that is desired

along the curve. Computation and clustering of grid points is obtained through an arc length redistribution along the grid curve followed by interpolation. The process is repeated for each grid curve.

5. CONCLUDING REMARKS

Based on our experience with the cranked-wing fighter configuration and other recent investigations (Refs. 12 to 14), it is presently feasible to generate composite block grids and compute Euler flow computations about complex three-dimensional configurations. The grid generation procedure that we and the investigators in reference 13 have used is totally algebraic. Other investigators have used a combination of algebraic and differential grid generation (Refs. 11,12 and 14). The grid generation software for our computations is experimental and aimed at one geometry type. Efforts are under way at Langley and elsewhere (Ref. 15) to generalize three-dimensional grid generation software.

Planning the block structure and topology is the most important grid generation step. Both the physical requirements (shocks, boundary layers, separation, etc.) and the solution technique requirements (singularities, skewness, etc.) must be resolved. Also, the alignment of grid curves with boundary surfaces can affect the accuracy of boundary conditions and the complexity of solution software. In our planning we have attempted to minimize the number of grid blocks to cover a domain and consequently simplify the software logic.

The configuration surface representation must be accurate and robust for extracting grid data. We have found that a large percentage of the overall effort must be devoted to surface representation and grid computation on the configuration surface.

Generating the surrounding grid for a three-dimensional configuration using transfinite interpolation is straight forward after proper planning and configuration surface grid generation. The primary aspects to keep in mind are the clustering of grid points and the continuity of grid curves across grid blocks. We have found that computer graphics is an essential tool in generating both the configuration surface grid and the surrounding grid. Our direction is to move toward doing these tasks in a workstation environment.

The finite-volume technique is very suitable for Euler flow computations on multiple-block grids. Solutions for the dual-block grid (264,000 grid points) about the fighter configuration can be obtained on the VPS-32 (CYBER 205) in less than one hour. Our experience with a finite-difference Navier-Stokes solver about a part of the fighter configuration is that it is extremely time consuming and appeared to be very grid sensitive. The availability of computer memory is adequate for Navier-Stokes solutions of complex three-dimensional grids, however, the CPU speeds are not adequate. In our case many hours were required to obtain only one solution over a part of a configuration. Consequently, we feel that we can make the most progress in the near future by pursuing geometric complexity of configuration surfaces and Euler flow computations.

6. REFERENCES

1. Smith, R. E.; Kudlinski, R. A.; and Everton, E. A.: Algebraic Grid Generation for Wing-Fuselage Bodies, AIAA Paper 84-0002.

2. Smith, R. E.; Kudlinski, R. A.; and Pitts, J. I.: Surface Grid Generation for Wing-Fuselage Bodies, NASA CP 2272, April 1983.

3. Craidon, C. B.: A Program for Fitting Smooth Surfaces to an Aircraft Configuration and Other Three-Dimensional Geometries, NASA TM X-3206, June 1975.

4. Smith, R. E.; and Eriksson, L. E.: Algebraic Grid Generation, First World Congress on Computational Mechanics, Austin TX., Sept. 1986.

5. Smith, R. E.; and Pitts, J. I.: The Solution of the Three-Dimensional Compressible Navier-Stokes Equations on a Vector Computer, Third IMAX International Symposium on Computer Methods for Partial Differential Equations, Lehigh University, PA, June 1979.

6. Smith, R. E.: Two-Boundary Grid Generation for the Solution of the Navier-Stokes Equations. NASA TM-83123.

7. Smith, R. E.; and Wiese, M. R.: Interactive Algebraic Grid-Generation Technique, NASA TP 2533, March 1986.

8. Eriksson, L. E.: Flow Solution on a Dual-Block Grid Around an Airplane, First World Congress on Computational Mechanics, Austin TX., Sept. 1986.

9. Eriksson, L.-E.; Smith, R. E.; Wiese, M. R.; and Farr, N.: Grid Generation and Inviscid Flow Computation About Cranked-Winged Airplane Geometries, AIAA paper 87-1125.

10. Eriksson, L. E.: Practical Three-Dimensional Mesh Generation Using Transfinite Interpolation, SIAM J. Fluid Mech., Vol. 148, 1984, pp 45-71.

11. Karman, S. L., Jr.; Steinbrenner, J. P.; and Kisielewski, K. M.: Analysis of the F-16 Flow Field by a Block Grid Euler Approach, 58th Meeting of the Fluid Dynamics Panel Symposium on Applications of Computational Fluid Dynamics in Aeronautics, Aix-En-Provence France, April 1986.

12. Fritz, W.; and Leicher, S.: Numerical Solution of 3-D Inviscid Flow Fields Around Complete Aircraft Configuration, 15th Congress International Council of the Aeronautical Sciences, London UK, Sept. 1986.

13. Sawada, K.; and Takanashi, S.: A Numerical Investigation on Wing/Nacelled Interferences of USB Configuration, AIAA Paper 87-0455.

14. Yu, N. J.; Chen, H. C.; Kusunose, K.; and Summerfield, D. M.: Flow Simulation for a Complex Airplane Configuration Using Euler Equations, AIAA Paper 87-0454.

15. Thompson, J. F.: A Composite Grid Generation Code For General 3-D Regions, AIAA Paper 87-0275.

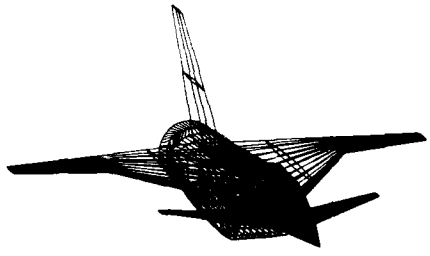


Figure 1.- Orthographic View of Cranked-Wing Aircraft.

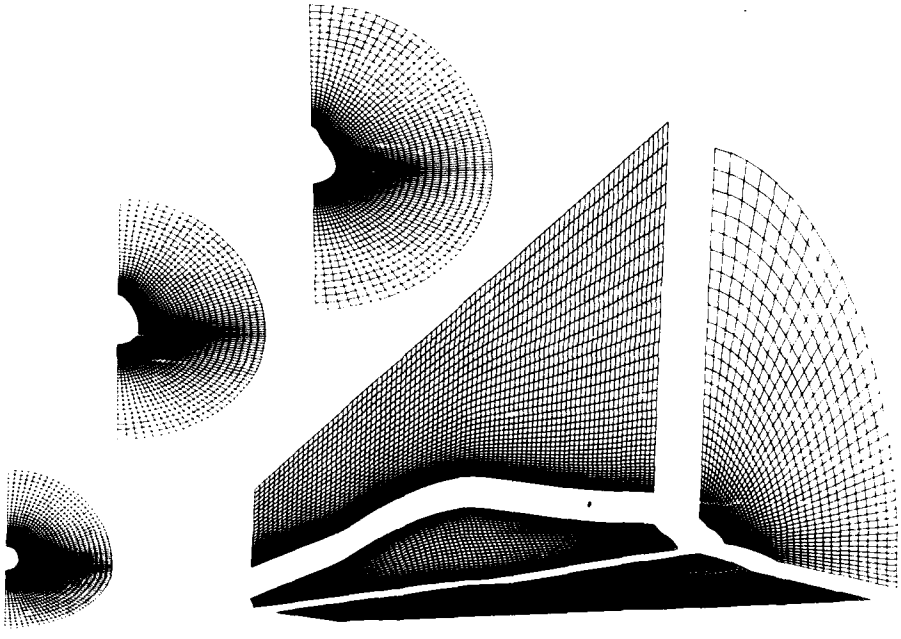


Figure 2.- Single-Block Grid-Forward Region.

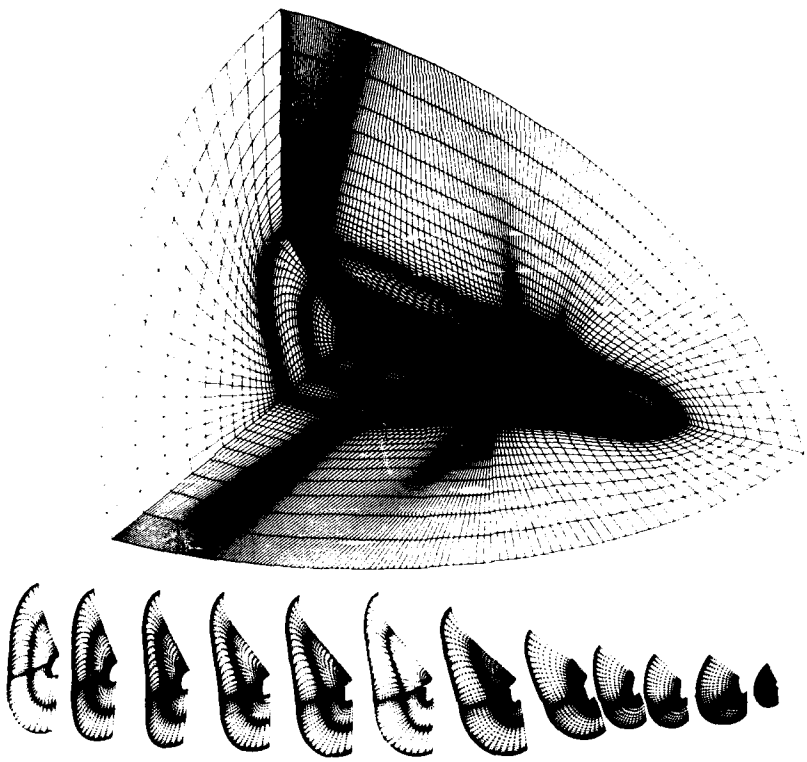


Figure 3 - Dual-Block Grid.

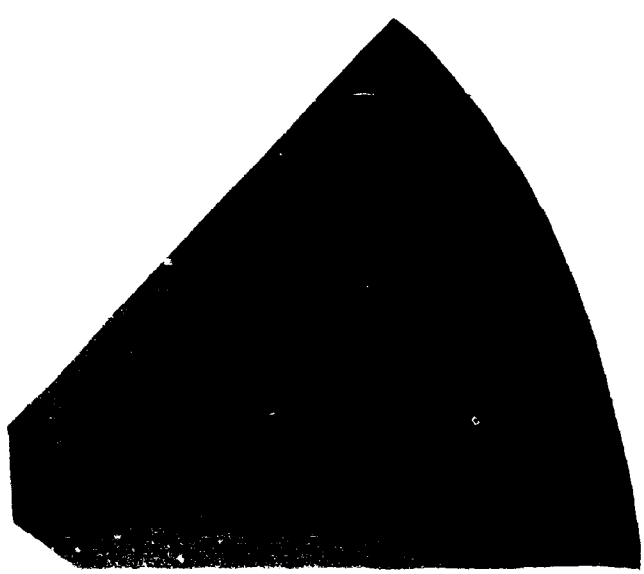


Figure 4.- Pressure Solution on Single-Block Grid.

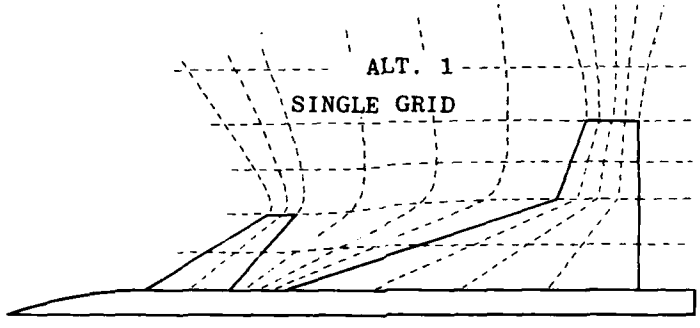


Figure 5.- Topology for a Single-Block Grid with Grid-Line Alignment with Lifting Surfaces.

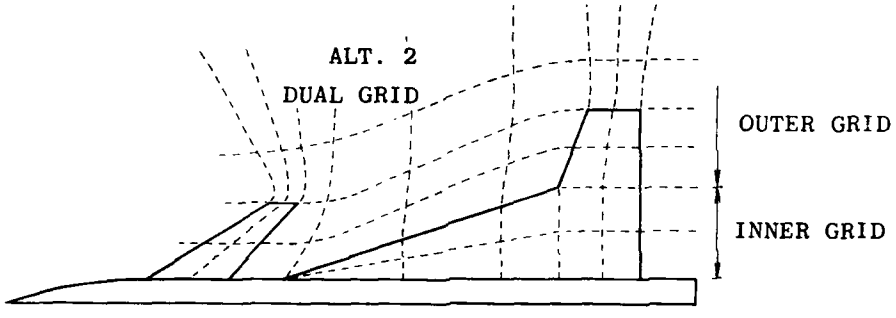


Figure 6.- Topology for a Dual-Block Grid with Grid-Line Alignment with Lifting Surfaces.

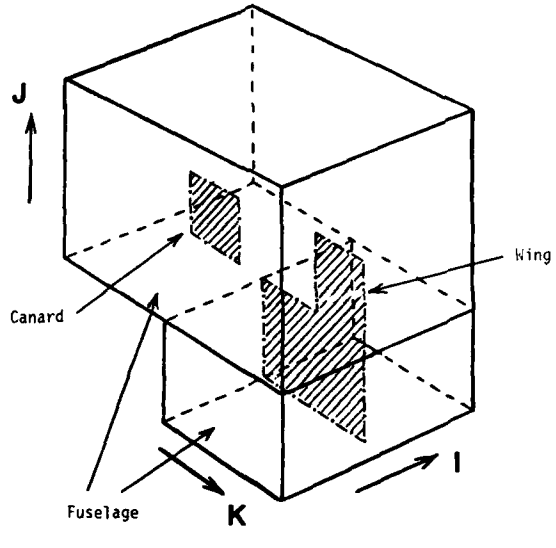


Figure 7.- Topology of Dual-Block Grid Relative to the Computational Domain.

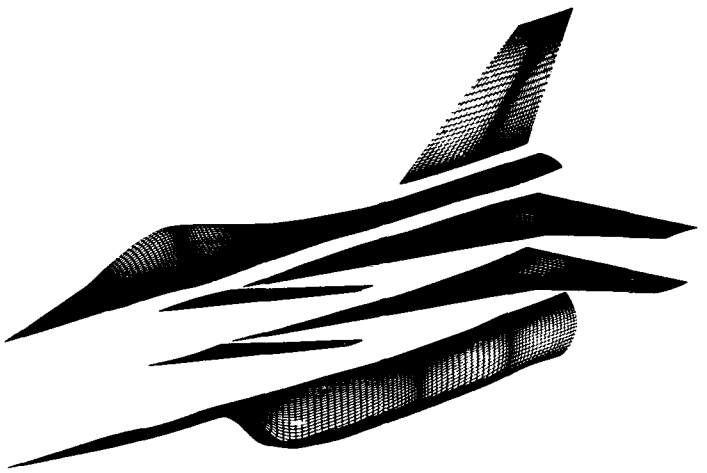


Figure 8.- Surface Grid in Exploded View.

Figure 9.- Pressure Coefficient on Selected Surfaces for Fighter Configuration.

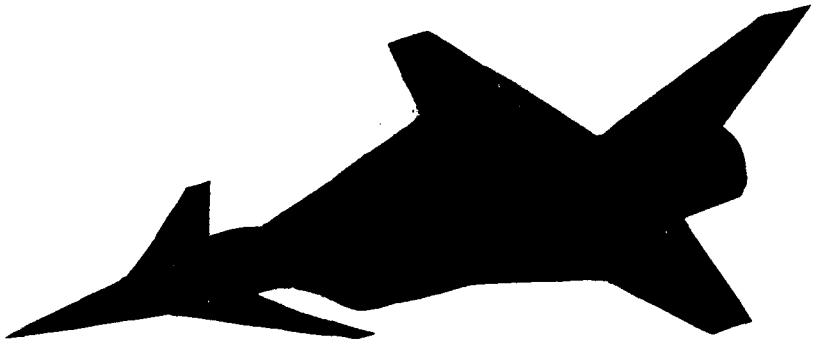


Figure 10.- Shaded Orthographic View of Fighter Configuration.

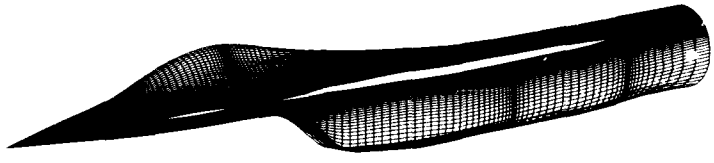


Figure 11.- Fuselage Grid and Intersections.

4.7 GRID GENERATION FOR AN ADVANCED FIGHTER AIRCRAFT

by
A. Eberle and W. Schwarz
Messerschmitt-Bölkow-Blohm GmbH
Helicopter and Military Aircraft Group
Theoretical Aerodynamics, LKE122
P.O. Box 80 11 60, D-8000 München 80, FRG

SUMMARY

The grid generation process for a realistic and complex fighter type aircraft will be described. The method is based on the solution of biharmonic equations and uses a single block concept. Only a few user specified input parameters are necessary for the construction of the space grid and therefore this grid generation system is very simple to handle. The grid is intended for calculations with an Euler code at transsonic and supersonic speeds.

1. INTRODUCTION

The automatic generation of computational grids around aerodynamic configurations becomes more and more important the more complex the body immersed in the fluid is. While the flow algorithm remains principally unchanged, the grid has to be adapted properly to every new geometry. Therefore, the grid generation process is the most time-consuming part of a flowfield calculation for a realistic and complex configuration, not in terms of computer time but in terms of man power.

A grid generation system for general, complex geometries should be reliable and simple to handle. This means that grids of reasonable good quality can be generated automatically with only a very small number of user specified parameters. From experiences with grid generators for automobiles, ducts, wings, wing-fuselage combinations and multi-finned missiles it seems that this can be done by using the biharmonic equation as a grid generation system. The whole grid construction is controlled only by the boundary conditions at the surface of the configuration and at the farfield boundaries. This formulation simplifies the procedure considerably and as far as we know, it is the simplest method reported so far.

To show the capabilities of this concept, the grid generation process for a realistic fighter type aircraft will be explained. The main features of this configuration are a fuselage with belly intake, cranked delta wing, canards and two lateral stabilizers. Because of computer storage limitations, a fine modelling of the intake region (boundary layer diverter, horizontal splitter plate and vertical plate, separating the left and right engine duct) and installed external stores are not included. This grid with approx. 500 000 points was subsequently used for several flow calculations with an Euler code at transsonic and supersonic speeds.

2. GRID STRUCTURE

The first step in every grid generation process is the choice of the grid topology best suited for the given configuration. While C-type or O-type grids are ideal for simple wings, it becomes more difficult or even impossible to treat complex geometries with this grid types. For this reason, we decided to use an H-type grid structure which is very flexible and can be adapted to very complex configurations by the use of interior branch cuts or by using a multi-block approach. Figure 1 shows the general lay-out of the grid where the outer boundaries are a simple rectangular box.

There are two different concepts for the generation of grids around complicated configurations. In a multi-block approach (cf. /1/, /2/, /3/) the entire flowfield is subdivided into a number of simple blocks and the grid is generated separately in every block. The main problem of this method is the treatment of the block boundaries. At the present time there seems to be no automatic scheme for the subdivision of the domain, and therefore the exact location of the block boundaries has to be evaluated in a time consuming trial and error process. To avoid these difficulties, we decided to use a single block grid structure. To resolve the complex configuration properly, several interior branch cuts had to be introduced. This leads to a number of singular points called lost or fictitious corners (cf. /4/) on the surface of the configuration. Figure 2 shows the structure of the surface grid in the computational space. We can see, that in this concept the configuration and not the whole flowfield , has to be divided into several blocks. The computational domain is a single block from which the cells, which lie inside the configuration, have to be excluded.

3. SURFACE GRID GENERATION

3.1 GEOMETRY INPUT

The input requirements are as simple as possible. Since the configuration is assumed to be symmetric about the plane y = 0, only one half is considered.

Fuselage:

The fuselage is given by wire frames expressed by coordinates x, y, z for each frame. Usually these frames are given at x = const. stations. In order to identify the intake geometry correctly, the fuselage is divided into two parts, the forebody and the afterbody (Figure 3). The last cross section of the forebody is partly identical to the first section of the afterbody. The latter consists of the last section of the forebody with the addition of the wire frame forming the intake.

Each cross-section is accompanied by two integers marking the lower and the upper fuselage block boundary points of the section. The lower block boundary line of the forebody ends at the last section of the forebody and should coincide there with the corner formed by the intake frame cutting point with the identical part of the fore- and the afterbody. The upper block boundary line should not have a jump at the common part of the fore- and the afterbody. The lower block boundary line of the afterbody should start at the corner of the box type intake. We decided to place the upper block boundary line outboard of the vertical V-stabilizer. Finally the coordinates x, y, z of the fuselage nose cap are given in random ordering by input. In the present case the fuselage is pointed, and the nose cap can be prescribed by a plane of a triple of coordinates, all with $x = x_{nose}$.

Lifting Surfaces:

It has proven practical to prescribe the planforms of the canard, the wing and the stabilizer first. The airfoil inputs are simply functions z(x). Each airfoil is accompanied by four numbers: the desired y-station in the planform, a twist angle, a point of rotation as fraction of cord length and an elevation Δz. Then the airfoils are adjusted to the given planform. An array of y-stations allows inserting additional airfoils which are generated from the linear interpolation of the two neighbouring airfoils. Finally the lifting surfaces are shifted and rotated to their final position at the fuselage. The final input wire model is shown in Fig. 4.

3.2 LIFTING SURFACES

The present goal is to obtain an equal number of coordinates for each airfoil of a lifting surface. Therefore a distribution of n abscissae for the lower and upper side of the airfoils is generated the following way.

a) Fix the nosepoint abscissa x_1 and the trailing edge abscissa x_n

b) Attract the new leading edge abscissa x_2 and the near trailing edge abscissa x_{n-1} to the nose respectively to the trailing edge by the formulas

$$x_2 = x_1 + a(x_3 - x_1) \tag{1a}$$

$$x_{n-1} = x_n + a(x_{n-2} - x_n) \tag{1b}$$

where 'a' is a global attraction parameter ($0 < a < 0.5$).

c) Calculate boundary sources

$$P_2 = x_1 + x_3 - 2x_2 \tag{2a}$$

$$P_{n-1} = x_{n-2} + x_n - 2x_{n-1} \tag{2b}$$

d) Calculate new abscissae by

$$x_i = \frac{1}{2}(x_{i+1} + x_{i-1} - P_i) \quad , \quad i = 3 \text{ through } n-2 \tag{3}$$

e) Calculate source distribution by

$$P_i = \frac{1}{2}(P_{i+1} + P_{i-1}) \quad , \quad i = 3 \text{ through } n-2 \tag{4}$$

f) Repeat step 2 through 5 until convergence

$$|x_{new} - x_{old}|_{max} < \epsilon$$

with ϵ being a small user specified number.

g) Interpolate airfoils at the new abscissae x; i = 2 through n-1, by a third order polynomial.

3.3 WING-FUSELAGE INTERSECTION

There remains to find the intersection of the lifting surfaces with the fuselage skin. Since most of the available CAD-systems are not capable to calculate the coordinates of the intersection from two pointwise defined surfaces an algorithm was included which does this work. It is based on solving the problem of finding the point of intersection of a straight line with a plane spanned by three coordinate triples in space.

For this purpose the fuselage wire frame model is interconnected contiguously by triangular finite elements (no overlaps, no gaps!). Upon that the intersection point of the quasi spanwise grid lines at near equal percentage with each of the linear triangular finite elements is calculated. If now one of the three baricentric triangle coordinates is negative, this triangle is discarded and the next one is taken for the intersection. Note that for each fuselage network there is only one triangle admissible for an intersection point. Finally the spanwise airfoil stations of the lifting surfaces are shifted by a prescribed ratio along the lines of equal percentage towards the fuselage surface to make them fit better with the curvature of the latter, see Figure 5.

3.4 FUSELAGE

The goal of the surface grid generation on the fuselage is mainly to cluster coordinates at the cuts formed by the intersections of the lifting surfaces with the fuselage. Figure 6 clears up the geometric situation, here for the fuselage side wall.

First the coordinates are attracted to the wing cut, e.g.

$$x_2 = x_1 + a(x_3 - x_1) \tag{5a}$$

$$z_2 = z_1 + a(z_3 - z_1) \tag{5b}$$

where 'a' is the global attraction parameter. Upon that the source boundary condition for the Poisson coordinate smoother is calculated, e.g.

$$P_2 = x_1 + x_3 + x_4 + x_5 - 4x_2 \tag{6a}$$

$$R_2 = z_1 + z_3 + z_4 + z_5 - 4z_2 \tag{6b}$$

The Poisson solver which generates the x,z side wall projection grid is

$$x_{i,k} = \frac{1}{4}(x_{i-1,k} + x_{i+1,k} + x_{i,k-1} + x_{i,k+1} - P_{i,k}) \tag{7a}$$

$$z_{i,k} = \frac{1}{4}(z_{i-1,k} + z_{i+1,k} + z_{i,k-1} + z_{i,k+1} - R_{i,k}) \tag{7b}$$

$$P_{i,k} = \frac{1}{4}(P_{i-1,k} + P_{i+1,k} + P_{i,k-1} + P_{i,k+1}) \tag{8a}$$

$$R_{i,k} = \frac{1}{4}(R_{i-1,k} + R_{i+1,k} + R_{i,k-1} + R_{i,k+1}) \tag{8b}$$

The steps
a) attraction
b) coordinate smoothing
c) boundary source calculation
are repeated till convergence. The Poisson operators are also applied along the block boundary lines such that the points on this lines float along them. The coordinates are linearly interpolated from the block boundary specificaton of the fuselage input.

After the fuselage projection grids are set up, the missing coordinate is interpolated from the fuselage input frames using linear triangular finite elements. At holes in fuselage geometry such as the intake and the nozzle exit, no surface interpolation is possible. In this case the Poisson smoother is applied on all three coordinates. It proved to be necessary to solve the Poisson equations simultaneously on all fuselage projection grids (bottom, ceiling, side wall, intake, nozzle) with the freedom that the grid points may float along the edges of the configuration, be they real edges or fictitious corner lines. In this particular way jumps in the curvature of the grid lines belonging to one family are brought to a minimum. Figure 7 shows the final surface grid with approx. 15 000 points.

4. SPACE GRID GENERATION

4.1 GENERAL DESCRIPTION

The grid generation for the flowfield follows the same concepts which were used for the distribution of the abscissae for the point distribution on the lifting surfaces and the generation of the two-dimensional projection grids forming the walls of the fuselage. As explained in Ref. /5/, the method is based on the solution of the biharmonic equation

$$\nabla^4 \vec{r} = 0, \tag{9}$$

which is actually implemented as a system of two second order partial differential equations (Poisson type)

$$\nabla^2 \vec{r} = \vec{S} \tag{10}$$

$$\nabla^2 \vec{S} = 0 \tag{11}$$

where $\vec{r}$ is the position vector

$$\vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

and $\vec{S}$ is the vector of the control functions (or source terms)

$$\vec{S} = \begin{pmatrix} P \\ Q \\ R \end{pmatrix}$$

After the introduction of central difference approximations for the second derivatives, equation (10) and (11) can be rearranged easily to yield the final expressions for the calculation of the coordinates $\vec{r}$ and the control functions $\vec{S}$ for a given node (i,j,k):

$$\vec{r}_{i,j,k} = \frac{1}{6}(\vec{r}_{i+1,j,k} + \vec{r}_{i-1,j,k} + \vec{r}_{i,j+1,k} + \vec{r}_{i,j-1,k} + \vec{r}_{i,j,k+1} + \vec{r}_{i,j,k-1} - \vec{S}_{i,j,k}) \tag{12}$$

$$\vec{S}_{i,j,k} = \frac{1}{6}(\vec{S}_{i+1,j,k} + \vec{S}_{i-1,j,k} + \vec{S}_{i,j+1,k} + \vec{S}_{i,j-1,k} + \vec{S}_{i,j,k+1} + \vec{S}_{i,j,k-1}) \tag{13}$$

Boundary conditions

Grid control is exercised via the boundary conditions for $\vec{r}$ and $\vec{S}$ at the inner boundary. This is done by the attraction of points towards the surface of the configuration as shown in Figure 8:

$$\vec{r}_2 = \vec{r}_1 + a(\vec{r}_3 - \vec{r}_1) \tag{14}$$

with the global attraction parameter 'a'.

With the help of eq. (11), this coordinates can be used to calculate the boundary values for the source terms at these points (cf. Fig. 8)

$$\vec{S}_2 = \vec{r}_1 + \vec{r}_3 + \vec{r}_4 + \vec{r}_5 + \vec{r}_6 + \vec{r}_7 - 6\vec{r}_2 \tag{15}$$

At the farfield boundary, an orthogonal intersection of the gridlines is imposed and the source strength is put to zero. At the symmetry plane, the point coordinates and the corresponding source strengthes are calculated by using a symmetry condition for equation (12) and (13).

For the numerical solution of this grid generation system, a simple Gauss-Seidel or SOR scheme can be used. Each iteration consists of the following steps:

a) Attraction of points towards the body surface (eq. 14),

b) Calculation of the new source terms at these points (eq. 15),

c) Calculation of the source terms in the flowfield (eq. 13),

d) Calculation of the point coordinates in the flowfield (eq. 12),

e) Orthogonal intersection of gridlines at the farfield boundaries.

4.2 SPECIAL REGIONS

Due to the complex structure of the configuration, several regions need a special treatment. The resolution of the region near the wing leading edge is very important for the accuracy of the subsequent flow calculations. For a sharp leading edge, it is sufficient to have a single grid point in the nose region as shown in fig. 9a. For wings with a blunt leading edge, this would lead to a grid with rather skewed cells in the nose region. In this case it is better to have more lines ending on the surface as demonstrated in fig. 9b. The same assessment is true for all similar edges on the surface of an aircraft, like wing tips and wing trailing edges. The wings, canards and stabilizers of the presented aircraft configuration have rather sharp leading and trailing edges and therefore a grid structure similar to figure 9a was used.

Another difficult region is the pointed fuselage nose. The grid in physical space has only one point in this region but in the computational space, there is a surface block boundary where about 250 grid lines end (cf. fig. 2). This means that all these lines have to end in one point (the fuselage nose) forming tetrahedral type grid cells in this region as shown in Figure 10.

Figure 11, 12, 13 show some surfaces of the resulting grid for the advanced fighter aircraft with approx. 500 000 points.

5. FLOWFIELD CALCULATIONS WITH BOX-TYPE H-MESHES

The grid used for the present aircraft code is in principle a single block grid with the body being carved out. This is a philosophy which differs from multi-block grids where after the surface specification of the configuration many blocks are fitted to the aircraft skin. The same philosophy also is traced in the implicit Euler code used for the flow calculations. The entire grid box is taken as a large 3D-DO-loop. The aircraft configuration is identified by a logical array saying 'no' for each dummy cell inside the aircraft and saying 'yes' in the cells where physical flow exists.

If a 'yes' ('no') follows a 'no' ('yes') then the code automatically sets the characteristic solid body boundary condition using the positive (negative) characteristic field for the flow value extrapolation to the boundary. This procedure is repeated three times, first for all i-lines, then for the j-lines and finally for the k-lines. The dummy cells inside the aircraft are included in the calculations in order to keep the vector lengths as long as possible. Each line algorithm is followed by the evaluation of the Euler flux differences. The flow variable update is performed by an implicit point Gauss-Seidel Newton type residual driver. Again the whole grid box is taken as a large 3D-DO-loop. This time, however, the DO-loop is performed twice in steps of two in order to avoid recursive formulae. The implicit solid body boundary condition is entered the same way as for the Euler flux differences.

This formulation - single block grid together with a single block Euler solver - seams to be the simplest way to treat the flow field past a complex configuration such as the present. The whole computer program consisting of

a grid generator,

a grid geometry plot software,

an Euler flux subroutine,

an implicit residual driver,

an Euler result plot software

contains 7 000 FORTRAN statements.

A more detailed description of this algorithm and several results from flow calculations are included in Ref. /6/.

6. CONCLUSION

The described grid generation system is a simple and reliable method for the construction of grids around complex configurations. The single block concept avoids the difficulties involved in the subdivision of the flow field into several blocks, which is necessary for the multi-block system. This problem has been reduced to the correct construction of a surface grid with the appropriate location of the block boundary lines on it. The disadvantage of the single block concept is the limited grid size due to computer storage limitations and furthermore it is not possible to introduce embedded and refined subdomains. But nevertheless the resulting grids should be sufficient for most applications, at least for Euler calculations.

Starting from the surface grid, the space grid can be generated easily by the use of the described biharmonic grid generation system. Although it is simple to handle because it requires only a few number of input parameters, there are some cases where you want to exercise more influence on the grid, at least in some regions. Therefore some kind of postprocessing would be desirable. This could include algebraic subdivision of cell layes (/6/) or even an interactive grid optimization concept (/7/).

7. REFERENCES

/1/ Karman, S. L., Steinbrenner, J. P., Kisielewski, K. M., "Analysis of the F-16 Flow Field by a Block Grid Euler Approach" in Applications of Computational Fluid Dynamics in Aeronautics, AGARD-CP-412, 1986, pp. 18-1 - 18-14.

/2/ Shaw, J., Forsey, C. R., Weatherill, N. P., Rose, K. E., "A Block Structured Mesh Generation Technique for Aerodynamic Geometries" in Proc. 1st Intern. Conf. on Numerical Grid Generation in Computational Fluid Dynamics, J. Häuser, C. Taylor (eds.), Pineridge Press, 1986, pp. 329 - 340.

/3/ Fritz, W., "Numerical Grid Generation around Complete Aircraft Configurations" in Applications of Computational Fluid Dynamics in Aeronautics, AGARD-CP-412, 1986, pp. 3-1 - 3-8.

/4/ Carr, M. P., Forsey, C. R., "Developments in Coordinate Systems for Flow Field Problems" in Numerical Methods in Aeronautical Fluid Dynamics, P. L. Roe (ed.), Academic Press, 1982, pp. 75 - 114.

/5/ Schwarz, W., "Elliptic Grid Generation System for Three-Dimensional Configurations Using Poisson's Equation" in Proc. 1st Intern. Conf. on Numerical Grid Generation in Computational Fluid Dynamics, J. Häuser, C. Taylor (eds.), Pineridge Press, 1986, pp. 341 - 352.

/6/ Eberle, A., Misegades, K., "Euler Solution for a Complete Fighter Aircraft at Sub- and Supersonic Speed" in Applications of Computational Fluid Dynamics in Aeronautics, AGARD-CP-412, 1986, pp. 17-1 - 17-12.

/7/ Seibert, W., "An Approach to the Interactive Generation of Blockstructured Grids Using Computer Graphics Devices", in Proc. 1st Intern. Conf. on Numerical Grid Generation in Computational Fluid Dynamics, J. Häuser, C. Taylor (eds.), Pineridge Press, 1986, pp. 319 - 328.

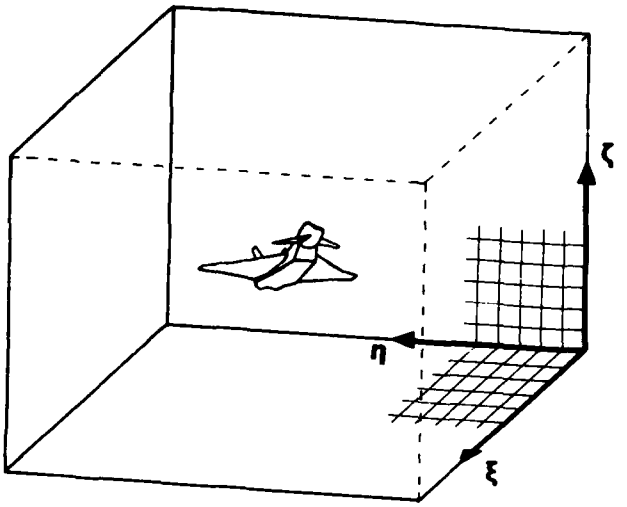


Figure 1: Grid Structure (Physical Space)

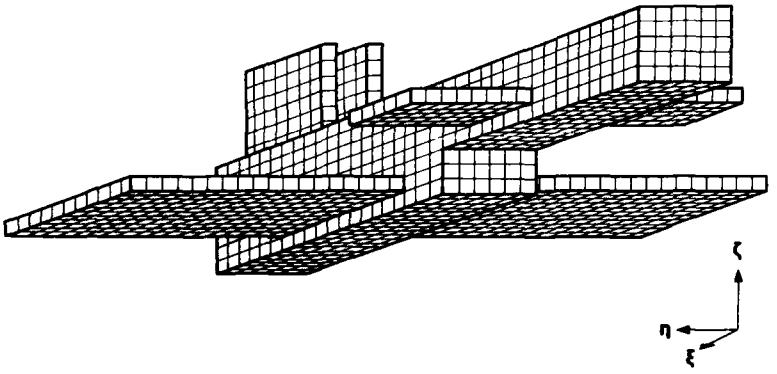
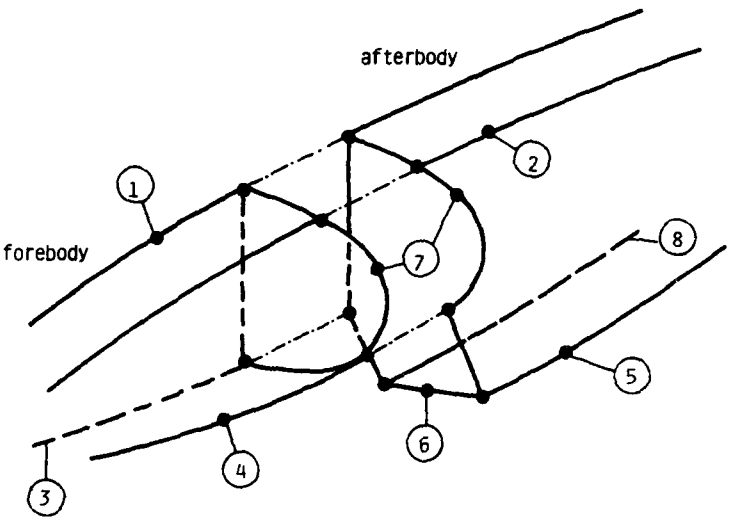


Figure 2: Surface Grid in Computational Space



① upper crown line
② upper fuselage block boundary
③ lower crown line of forebody
④ lower forebody block boundary
⑤ lower afterbody block boundary
⑥ intake wire frame
⑦ identical cross sections of fore- and afterbody
⑧ lower crown line of afterbody

Figure 3: Connection of Fuselage Fore- and Afterbody

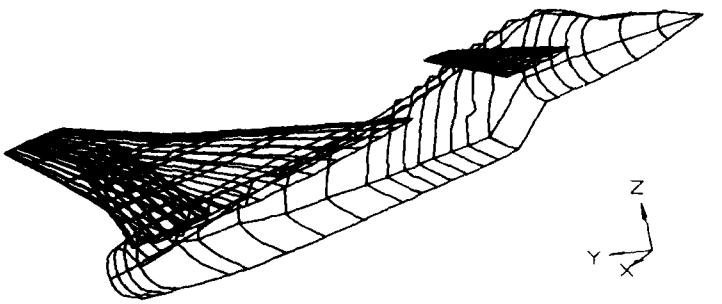


Figure 4: Complete Surface Geometry Input

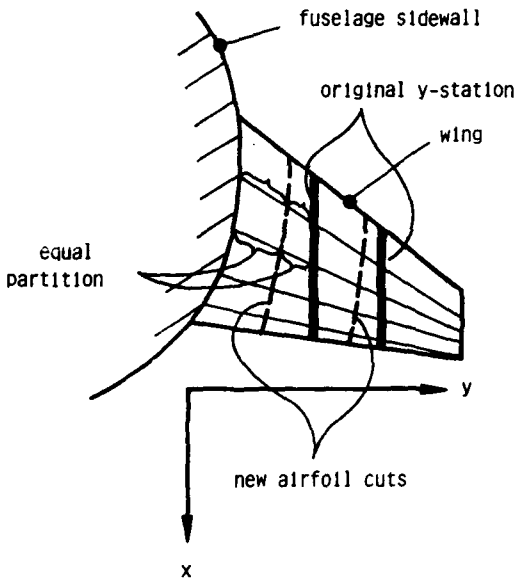


Figure 5: Wing-Fuselage Intersection

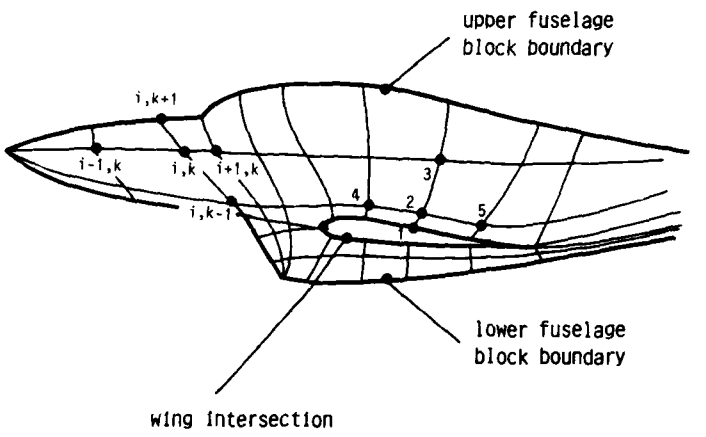


Figure 6: Surface Grid Generation on Fuselage Sidewall

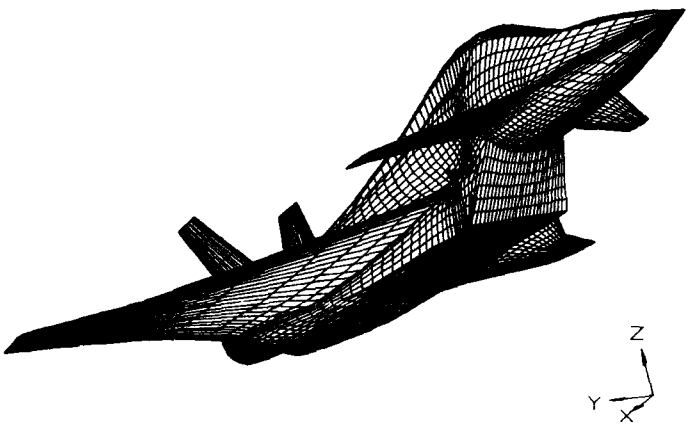


Figure 7: Surface Grid

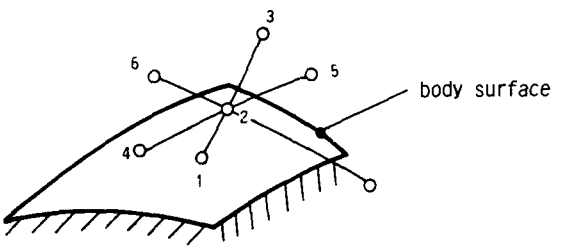


Figure 8: Attraction of Grid Points towards the Surface of the Configuration

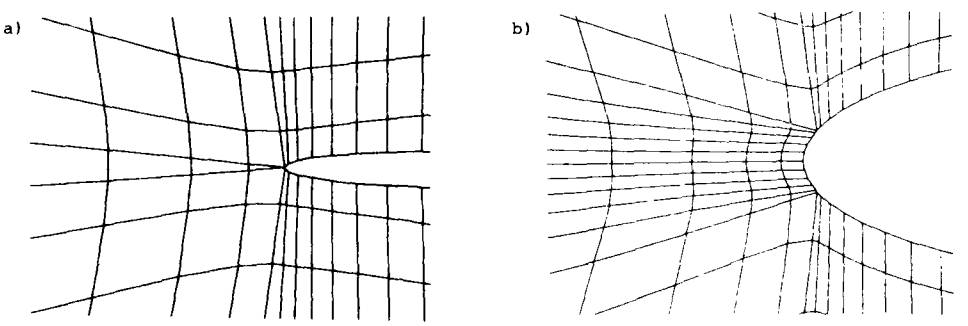


Figure 9: Different Grid Structure for Airfoils with
a) Sharp Leading Edge
b) Round Leading Edge

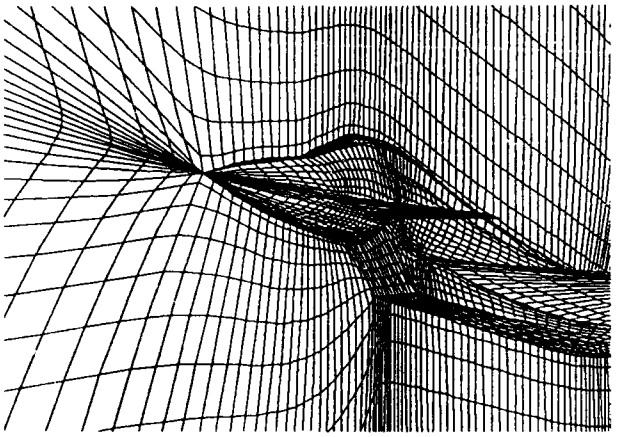


Figure 10: Details of Mesh near Fuselage Nose and Intake

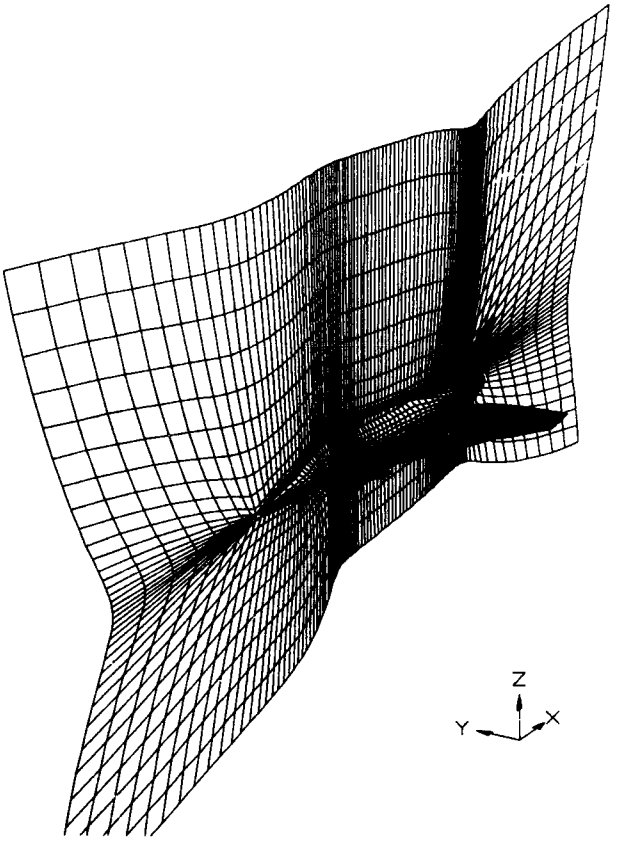


Figure 11: Surface j = const. (Symmetry Plane)

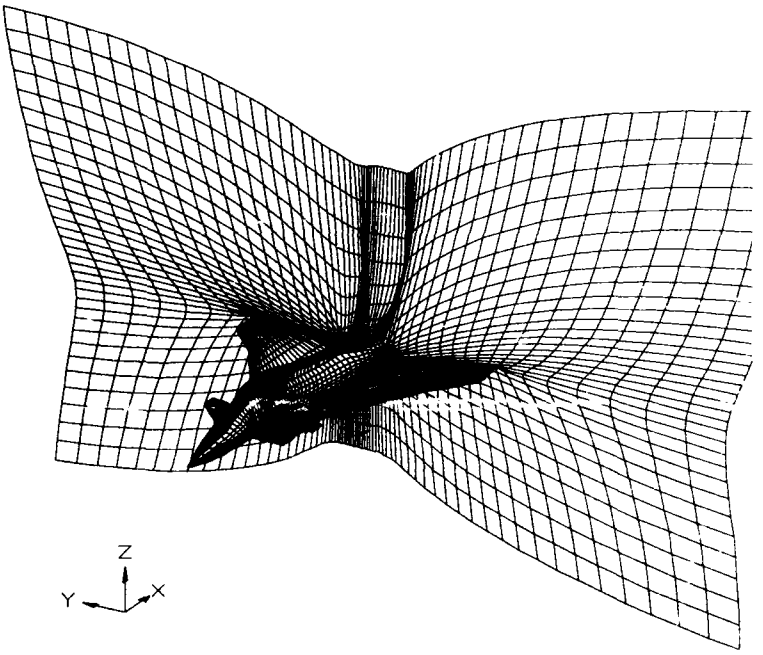


Figure 12: Surface i = const.

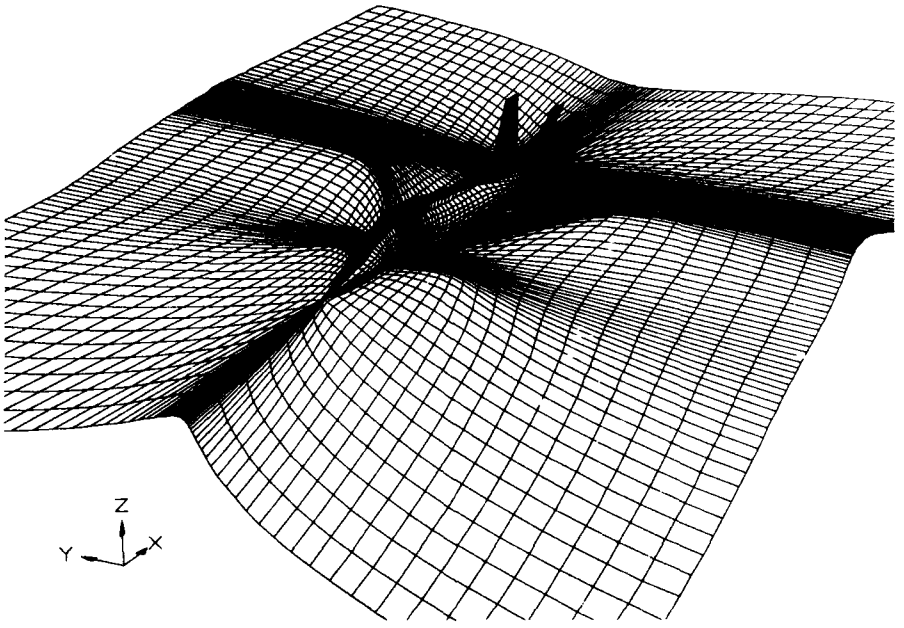


Figure 13: Surface k = const.

4.8

Algebraic Grid Generation for Fighter Type Aircraft

by

John Steinhoff

Department of Engineering Science and Mechanics
The University of Tennessee Space Institute
Tullahoma, Tennessee 37388

A systematic procedure is presented for synthesizing a complex computational grid for fighter type aircraft out of a number of simpler "elementary" grids. This method is useful when a grid is required over an object which, though complex, consists of a number of simpler pieces, such as an aircraft with a number of lifting surfaces. The procedure presented allows a smooth complex grid to be generated which becomes exactly equal to each elementary grid as the surface corresponding to that elementary grid is approached. In this way, methods which may have previously been developed for each piece do not have to be changed and can be used as "black boxes," whether they are algebraic, partial differential equation based, or whether the grids are just given numerically. This blending technique is only one of several tools which we use to generate effective grids. Other techniques include projection methods for generating surface grids. Some advantages and limitations of the method are discussed and examples are given of its use in generating complex fighter grids.

1. INTRODUCTION

For many aircraft geometries, the computational domain can be decomposed into a number of pieces each of which is fairly simple. Also, often an adequate grid can be easily generated for each of these pieces, if considered by itself. Our basic method involves blending these "elementary" grids into one smooth composite grid. This technique can be used over an entire aircraft, where simple methods exist for generating grids individually over each of the lifting surfaces and the pieces of the body. An important feature of the concept is that it can be used recursively: Composite subgrids can first be formed from elementary grids, using the method. Then, the same method can be used to form larger composite grids out of these individual subgrids. If algebraic methods are used to form each elementary grid, which can often be done since each piece is simple, then the entire grid generation procedure is algebraic, since the blending is non-iterative and involves no partial differential equation solutions. Accordingly, where applicable, it is a fast method suitable for interactive use. Also, if a partial differential equation is to be solved for some physical quantity and an iterative method is used to solve a set of discrete equations on the grid, which is usually the case, then at each iteration the grid can be quickly regenerated and there is no need to store the entire grid system. This feature can be especially important for large three-dimensional problems. This method is very different from other algebraic methods, such as those of Eiseman [1]. Each elementary grid is taken to be previously determined, either by algebraic methods, partial differential equation solution [2], or any other means. These grids can be defined over the entire space, rather than just on surfaces as in "transfinite interpolation" schemes.

An important feature of the method is that it allows the grid designer to use software packages and methods already developed or being developed by others (which can be quite sophisticated and complex) for the elementary grids about each piece of the problem. These can be used as "black boxes", and after each elementary grid is generated the grid designer can blend them together. Also, after a composite, complex grid is generated, if one of the pieces is later modified, only the single new elementary grid need be recomputed and blended into the composite grid.

Since the method is local, and each piece only influences the grid in its vicinity, local methods of controlling the grid can be formulated. This could be required, for example, if resolution were inadequate or if grid lines

were to cross. It will be seen that advantages of the method include simplicity and speed, even for complex geometries. Disadvantages include the lack of guarantees against line crossing (although this can be made unlikely), possible skewness (although this can be corrected) and the requirement that each elementary grid locally have the same topology. Also, for the three dimensional problems described here, other techniques are required which include projection of the wing grids onto the body surface.

2. BASIC METHOD

The basic grid generation method involves the blending technique of Ref. 1, described in the next section. Generation of a grid is recursive: First, a 3-D grid is computed about the aircraft body alone (denoted G_1). Then, an "elementary" grid is developed in a region about the wing (denoted G_w). This is then blended with the body (G_1) grid to form a smooth wing-body grid (G_2). As we approach the wing surface in grid G_2 in computational (i, j, k) space, the values $(x, y$ and $z)$ of the physical coordinates of the nodes smoothly approach the values of the elementary grid (G_w) coordinates, and are exactly equal to them on the wing surface. As we approach the boundary of the grid G_w in computational space, the composite (G_2) grid coordinates approach the original grid coordinates (G_1). In the wing-body junction region the elementary G_w grid is projected onto the body surface and then blended.

Other elements of the aircraft are similarly incorporated in a recursive way: the canard is first used to generate an elementary canard grid G_c. This is then blended with the wing-body grid, G_2, to form the wing-body-canard grid G_3. The blending is such that as we approach the canard in computational space, the grid approaches G_c. The tail is similarly incorporated by blending an elementary tail grid, G_t, with G_3 to form the final grid, G_4.

The basic topology is cylindrical about the body, as shown in Fig. 1. In each cylinder-like surface the lifting surfaces are mapped using special "H" grids. These H grids are singularity-free. A detailed study of the accuracy of these grids for airfoils for compressible flow computations was presented in Ref. 2 and found to be comparable to conventional "O" and "C" grids.

A very important feature of our data management is that the geometric data defining each element is kept in a separate file. Each set has the same format except for the body, which is only slightly different. This has led to considerable simplifications. For example, simple grids can easily be generated for diagnostic purposes by incorporating only one of the elements at a time.

3. BLENDING TECHNIQUE

Consider a set of N grids, each spanning the same computational space and approximately the same physical space. For simplicity, we define the computational coordinates to be just the (integer) indices of the grids. Thus, in n dimensions we have an n component vector, $\mathbf{r}_m(\mathbf{l})(\equiv (x_m(\mathbf{l}), y_m(\mathbf{l}), \quad z_m(\mathbf{l}))$ for $n = 3$) defined on each grid (labeled m) as a function of the indices $\mathbf{l}(\equiv (i, j, k)$ for $n = 3)$. It is important to think of the n components of $\mathbf{r}_m$ as ordinary smooth functions defined in the computational ($\mathbf{l}$) space. Defining non-negative weighting functions $P^m(\mathbf{l})$, the physical coordinates of the composite grid are then simply weighted sums of those of the elementary grids:

$$\mathbf{r}_c(\mathbf{l}) = \sum_m P^m(\mathbf{l})\mathbf{r}_m(\mathbf{l}) \Big/ \sum_m P^m(\mathbf{l}) .$$

The weighting functions are, in general, functions of all of the indices $\mathbf{l}$, and are a function of how close in computational space the point $\mathbf{l}$ is to the elementary surface segments. When $\mathbf{l}$ approaches some surface segment, say m_1, then $P_{m_1}(\mathbf{l})$ must approach 1 and all the other P's must approach 0 since there we must have

$$\mathbf{r}_c(\mathbf{l}) \to \mathbf{r}_{m_1}(\mathbf{l}).$$

Some of the "art" of using the method resides in the determination of the functions $P^m(\mathbf{l})$. Since values of $\mathbf{r}_m(\mathbf{l})$ which define smooth grids are determined separately about each elementary surface, the $P^m(\mathbf{l})$ do not

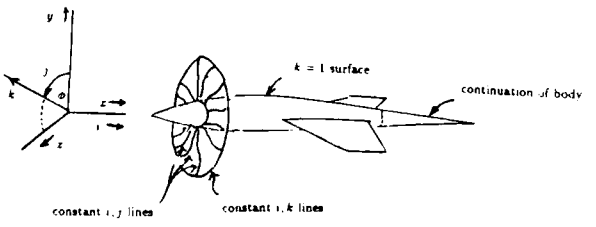


Figure 1 Definition of Grid Topology

have to do as much work as in an interpolation method where they typically completely determine one of the coordinates. It will be seen that very simple functions are sufficient. The main problems arise when grids must be blended with very different values of $\mathbf{r}$ in certain regions of $\mathbf{l}$ near an elementary surface. Then, care must be taken that a number of derivatives of $P^m(\mathbf{l})$ are 0 as $\mathbf{l}$ approaches the elementary surface (m_1), in addition to the value of $P_{m_1}(\mathbf{l})$ approaching 1. As more derivatives are made to go to 0, the region in $\mathbf{l}$ space where $\mathbf{r}_c(\mathbf{l})$ approaches $\mathbf{r}_{m_1}(\mathbf{l})$ becomes larger.

We choose a distance function from point $\mathbf{l}$ to each segment.

$$\tilde{z}^m = [(max(0, i - i_2^m, i_1^m - i))^2 + (max(0, j - j_2^m, j_1^m - j)) + (max(0, k - k_2^m, k_1^m - k))^2]^{1/2}$$

where we take the segment to lie in the region

$$i_1^m \le i \le i_2^m \; ; \; j_1^m \le j \le j_2^m \; ; \; k_1^m \le k \le k_2^m.$$

Each $\tilde{z}^m$ vanishes on segment m. We define a "global" distance function, z^m, for each segment (m) that is 1 when $\mathbf{l}$ approaches the segment $(\tilde{z}^m \to 0)$ and 0 when $\mathbf{l}$ approaches any other segment $(\tilde{z}^{m'} \to 0, m' \neq m)$:

$$z^m = \frac{1/\tilde{z}^m}{\sum_k 1/\tilde{z}^k}$$

Then, we simply have

$$P^m(\mathbf{l}) = \frac{1}{2}|1 - cos(\pi z^m)|.$$

With this definition the weighting functions approach the correct values on the segments and the first derivatives vanish.

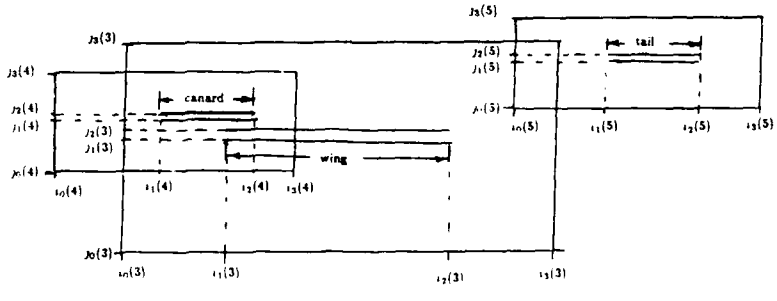
In two dimensions the method, as described, is applicable even if the boundary segments are contiguous. In three (and higher) dimensions the method is directly applicable only if the boundaries are not contiguous. In the fighter geometry treated in this paper, some of the pieces (such as wing and body) are joined. In the junction region, the surfaces of one grid (wing) have to be projected onto the surface of the other (body) before they can be blended.

The particular type of blending region that we have for the aircraft grid involves three dimensional rectangles in (i, j, k) space representing the region where the new elementary grid is generated. Embedded in this region are flat rectangular surfaces with one of the indices (j) fixed, which correspond to the lifting surfaces. A set of typical cross sections in (i, j) space is shown in Fig. 2. There are two types of surface; On one, representing the new lifting surface that we are adding into the grid, the composite grid coordinate (x, y, z) values must approach the new elementary values. The other surfaces represent either lifting surfaces already added to the previous grid which intersect the region of the new grid, or the outer boundaries of the elementary grid region. On all of these surfaces, the composite grid coordinate values must approach the previous grid values, so that previously included surfaces are not changed by the inclusion of the new surface. Thus, the new lifting surfaces correspond to one set of coordinates (the new elementary grid) and all others correspond to the original grid from the previous step.

4. SEQUENTIAL GRID GENERATION

The grid is generated in a sequence of steps, each of which incorporates another element of the fighter, while keeping the same topology, as follows:

Figure 2 $i - j$ Index Structure for Grid



4.1 Body

A "body" grid (G_1) is generated which conforms to the body at $k = 1$ between two i values ($i_1(1)$ and $i_2(1)$). The entire grid is defined to lie in the region

$$i_0(1) \le i \le i_3(1) \; ; \; j_0(1) \le j \le j_3(1) \; ; \; k_1 \le k \le k_3(1).$$

We take

$$i_0(1) = j_0(1) = k_0(1) = 1.$$

This mapping involves defining a set of (i, j) lines which emerge from the body on constant $-x$ surfaces, but normal within those surfaces at angle ϕ'_1, and change azimuthal angle ϕ according to

$$\phi'_k = \phi'_1 \times f_k + \phi'_{k_3} \times (1 - f_k)$$

where

$$\phi'_{k_3} = \pi\left(\frac{j-1}{j_3(1)-1}\right)$$

and the weighting function

$$f_k = \frac{1}{2\left[1 - \cos\left(\pi \frac{k-1}{k_2(1)-1}\right)\right]}$$

The radial (r) coordinate of each point is a simple function of k and the initial radius at $k = 1$, and the axial, or x value is chosen to approximately match the mean sweep of the wing.

4.2 Main Wing (Wing 1)

This step involves first computing an elementary three dimensional grid about the main wing and then blending it smoothly into the existing body grid (G_1). The wing is assumed to be defined at "span stations" or constant z_s planes, where z_s is the distance along a rotated z axis approximately aligned with the wing. First, the G_1 grid is rotated so that the wing is along the z axis. Then, G_1 is stretched and sheared to a new coordinate system where the wing has a constant cross section in the root region (see Figure 3). With this transformation, a single 2-D airfoil grid can be used in this entire region. This grid is defined in an $x - y$ plane using a 2-D H-grid method described in Ref. (2). This grid has been developed for accurate airfoil solutions and is designed to be singularity-free at the leading edge by analytically removing the singularity there. This 2-D grid is first blended into the G_1 grid as described below, and then projected onto the constant-k surfaces of the rotated and stretched G_1 grid in the root region. The stretching and rotation are then reversed so that the original coordinate system is restored. The wing grid is then projected and blended onto the other k - surfaces in the root region. The exact z values obtained above are used for $k = 1$ so that the exact body shape is retained but now with a wing-fitted coordinate system. The values of z for subsequent k values are weighted sums of these computed, projected z values and blended z values, so that at the end of the root region, and beyond, each 2-D wing grid surface in the final system for this step, is at constant z. This allows us to have more control over the wing grids away from the body. The results of these operations is the grid G_2.

The blending is accomplished as described above and in Ref. (1.): We have our projected "elementary" wing grid system defined in a region of i, j, k space around the wing:

$$i_0(L) \le i \le i_3(L), j_1(L) \le j \le j_3(L); k_0(L) \le k \le k_3(L)$$

where $L = 2, 3, 4$ denotes either the main wing, canard, tail or other surface. Each wing is defined by

$$i_1(L) \le i \le i_2(L) \; ; \; k_1(L) \le k \le k_2(L)$$

with $j = j_1(L)$ for the lower surface and $j = j_2(L) = j_1(L) - 1$ for the upper surface. We let the values of x, y (and z outside the root region) at each node in the final grid for this step (G_2) be a weighted sum of the x, y

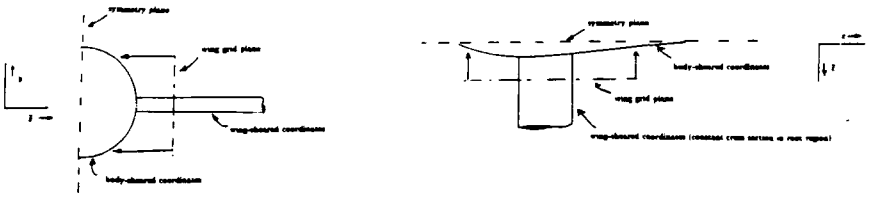


Figure 3 Wing Grid Projection Onto Body

and z values at the same node for G_1 and the wing grid described above. This weighting becomes zero for the wing grid (1 for G_2) at the outer boundaries of the wing grid and 1 (0 for G_2) at the wing surface. The result of this blending and projection is a smooth grid containing the body and first wing. The first wing is typically the main wing ($L = 2$), $L = 3$ corresponds to the canard and $L = 4$ to a tail.

4.3 Canard (Wing 2).

The generation of the elementary canard grid and blending into the main grid is done exactly as the wing in step 2. The fact that the main grid now has a wing makes no major difference, even if it intersects the box defining the elementary canard grid. It also makes no major difference if the canard intersects the original wing grid box. In fact, the original wing grid box boundaries are not used after step 2. If the wing intersects the canard box we merely have an additional constraint on the blending function: the weighting function for the canard grid *must vanish on the wing surface where the weighting function* for the main grid (now G_2) must approach 1. The distance and weighting functions are structured to easily accommodate these constraints for an arbitrary number of surfaces (see Figure 2)

4.4 Tail (Wing 3)

This step, as well as any subsequent ones involving additional surfaces, is done in the same way as above, where all previous surfaces that intersect the new elementary grid box are taken into account and the elementary grid weighting function made to vanish there. Again, no constraints are imposed on possible intersections of the various elementary grid boxes.

5. RESULTS

In Figure 4 a top view of the total configuration for a typical "generic" fighter is shown, with surface lines of constant j (body) or k (wings) depicted. It can be seen that the first wing sections conform to the body and gradually conform to constant z planes as the tip is approached. In Figure 5 a side view of the configuration is presented.

The surface grid on the body ($k = 1$) is shown in side view in Figure 6 and the grids on shells $k = 3$ and $k = 5$ are presented in Figures 7 and 8. The last can be seen to be near the tip of the canard. Figures 9, 10 and 11 present the same surfaces but rotated by 45° about the body axis to depict the tail region.

Figure 12 depicts an unblended ("elementary") main wing grid for shell 1. This is generated independently of any other elements, as are the canard and tail grids.

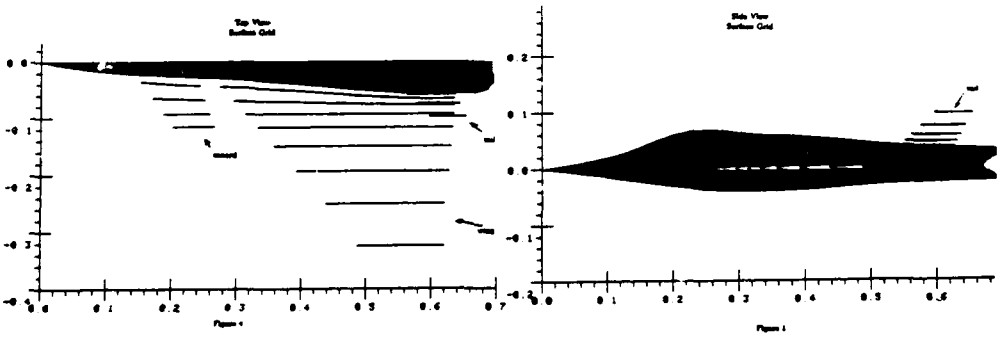
Figures 13 and 14 present top view of $j = 28$ and $j = 34$ surfaces, which contain the lower surfaces of the main wing and canard, respectively. The constrained outline of each can be seen.

Figures 15, 16 and 17 depict a front view of a contant $-i$ plane that contains the canard, wing, and wing/tail respectively. These elements can clearly be seen.

The grid depicted in the above figures contains approximately 173,000 points. Its generation required approximately 16 minutes on a VAX 11 785 minicomputer.

A grid similar to that described above, but with bunching near the lifting surfaces was used in a potential flow code to generate a subsonic solution. The code involved a conservative finite volume difference scheme and an ADI solution method.

A configuration similar to the EFA was treated next. The inlet was plugged giving a smooth body surface. The side view of the body surface grid is presented in Figures 18 and 19. The side view of shell number 16 near the wing is presented in Figure 20. Constant j-planes containing the wing ($j = 26$) and canard ($j = 32$) are presented in Figures 21 and 22, respectively. In Figure 23 a constant-i section near the nose cutting the



canard is presented. Figure 24 depicts a constant-i section near the middle and cutting the wing. In Figure 25 a similar section is shown, but near the aft section cutting both the wing and (vertical) tail. It can be seen in this figure that the wing is low on the body and almost tangential at the junction. In spite of this, the wing grid projection method was able to treat this case and map an adequate surface grid onto the body.

6. CONCLUSIONS

The blending method together with the projection technique appears to offer a relatively simple, fast and economical method of generating complex grids. The method has been implemented in a computer code, **HPLANE**. As yet the selection of the some indices of the elements has not been automated, but the code is still relatively easy to use for configurations similar to the one presented. A number of configurations have been treated including forward swept wings and cases with canards near an inlet "shelf" such as the JAS-39. Presently, it appears that smoothly varying canards, wings and tails can be handled by adjusting the input data to our present code. Other features such as fillets, discontinuities in lifting surfaces and inlets require special treatment. Although a combination of blending, projection and ordinary shearing can apparently still be used successfully to generate good grids, these features are highly individualized and some new programming is needed for new cases. A very importanat feature appears to be the ability to generate new grids quickly and cheaply, so that changes can be implemented in a short amount of time. The modularity utilized in our method, together with the algebraic approach accounts for this. With more experience with a number of different geometries, it may be possible to develop a single, general code for a wide range of configurations.

7. REFERENCES

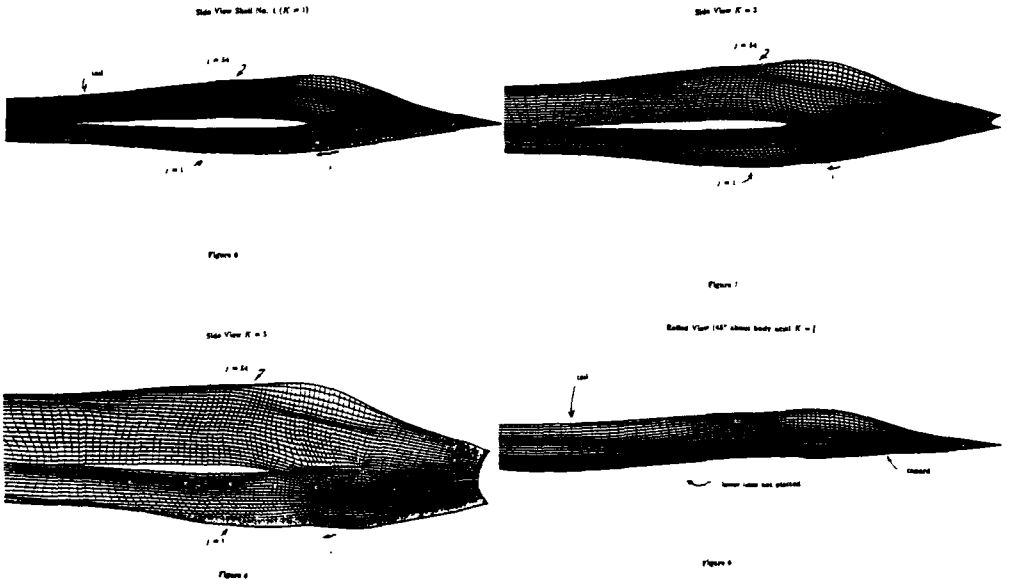
1. J. Steinhoff, "Blending Method for Grid Generation," to be published in Journal of Computational Physics, 1986.

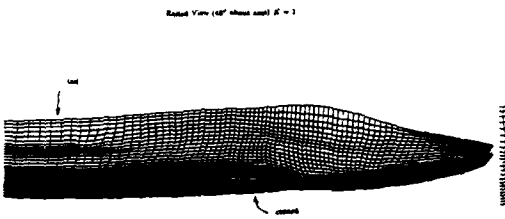
2. R.B. Pelz and J.S. Steinhoff, "Multigrid-ADI Solution of the Transonic Full Potential Equation for Airfoils Mapped to Slits." *Computers in Flow Predictions and Fluid Dynamics Experiments*, Proceedings of the ASME Winter Meeting, Washington, D.C., November, 1981, pp. 27-34.

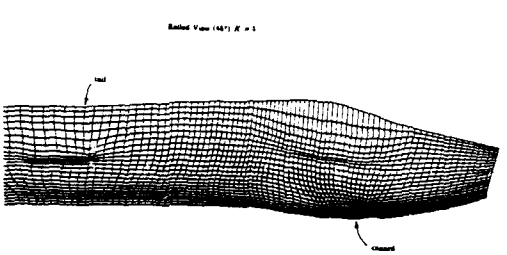
3. P.R. Eiseman, "Grid Generation for Fluid Mechanics Computations," vol. 17, *Annual Review of Fluid Mechanics*, Annual Reviews, Inc., Palo Alto, CA, 1985, pp. 487-522.

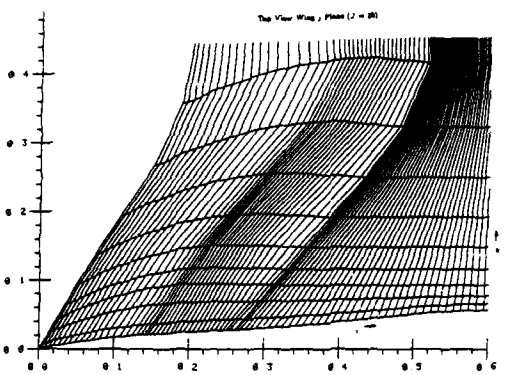
Acknowledgment

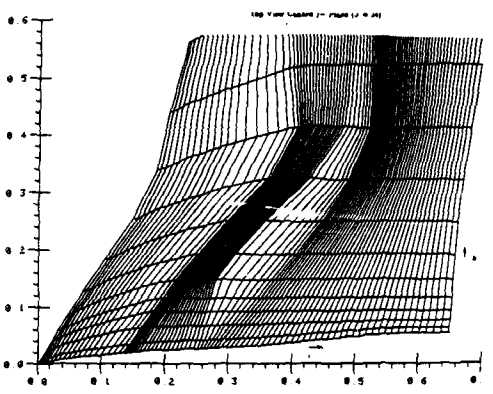
Some of this work was supported by Dornier, GmbH. Mr. H. Senge and K. Ramachandran assisted in generating the figures.

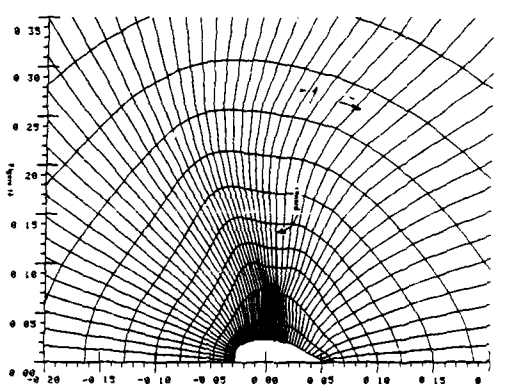


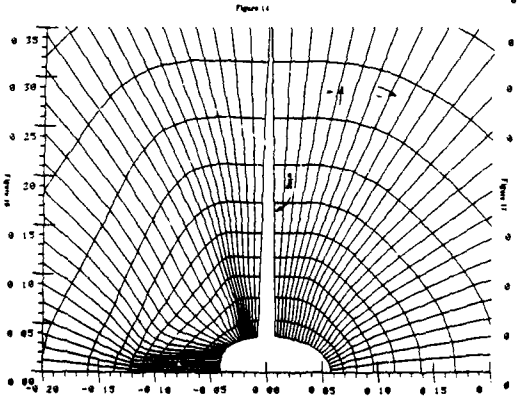


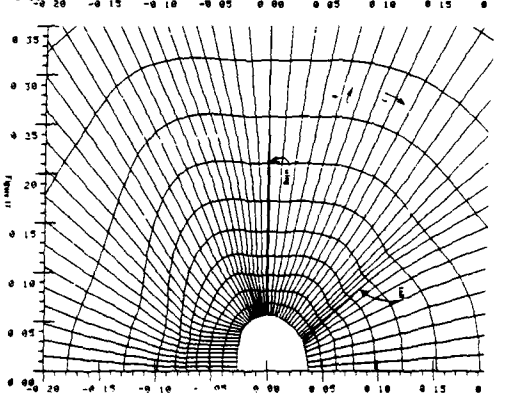


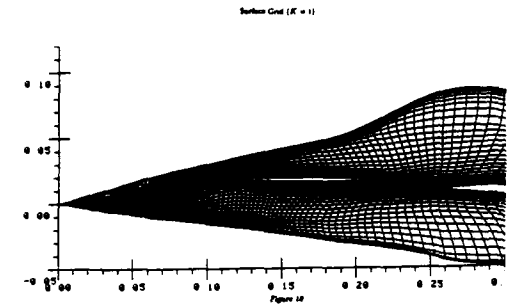


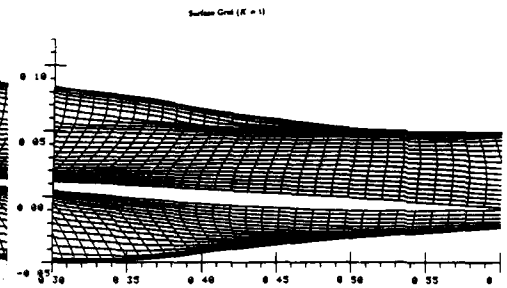


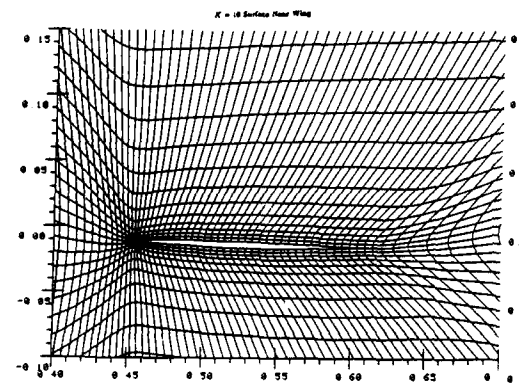


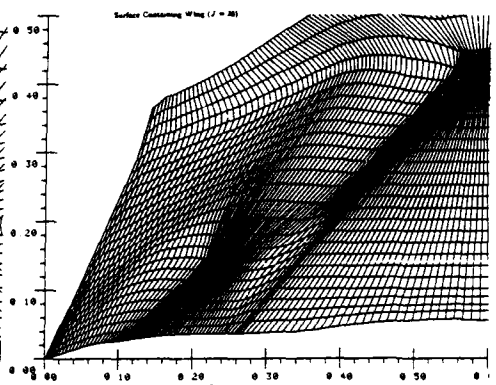


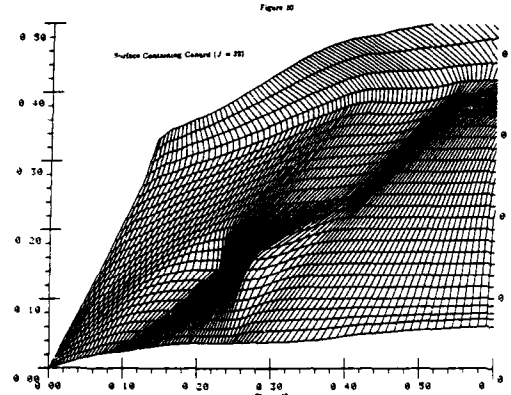


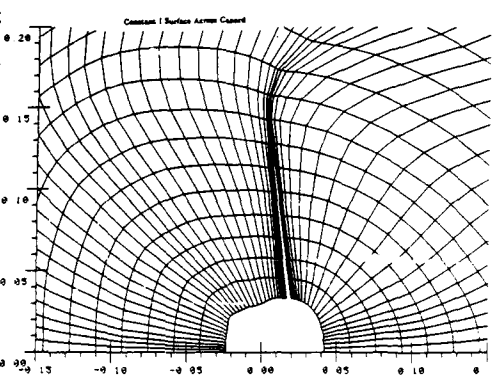


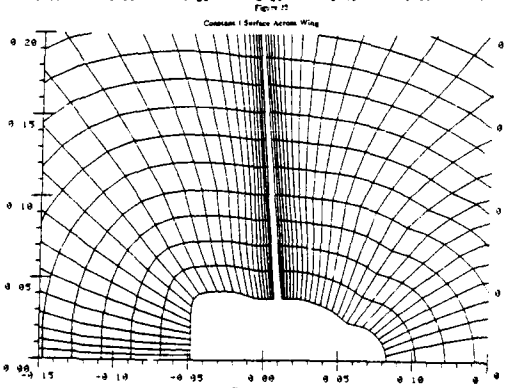


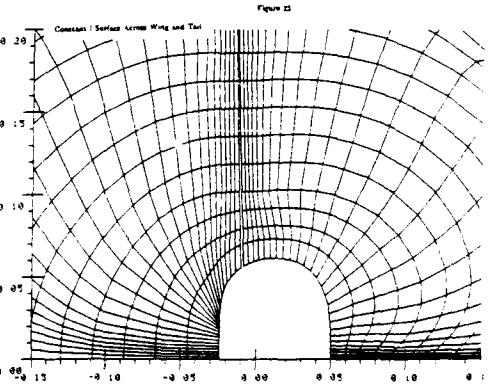












4.9 Composite Grid Generation for Aircraft Configurations with the EAGLE Code

Joe F. Thompson
Department of Aerospace Engineering
Mississippi State University
Mississippi State, MS USA

Lawrence E. Lijewski
U.S. Air Force Armament Laboratory
Eglin AFB, FL USA

SUMMARY

A general three-dimensional grid generation code based on a composite block structure is discussed. The code can operate either as an algebraic generation system or as an elliptic generation system. Provision is made for orthogonality at boundaries and complete continuity at block interfaces. The code can operate in two or three dimensions, or on a curved surface. The input is structured to be user-oriented, and arbitrary block configurations can be treated.

INTRODUCTION

The construction of computational fluid dynamics (CFD) codes for complicated regions is greatly simplified by a composite block grid structure since, with the use of a surrounding layer of points on each block, a flow code is only required basically to operate on a rectangular computational region. The necessary correspondence of points on the surrounding layers (image points) with interior points (object points) is set up by the grid code and made available to the CFD solution code.

The present grid code, developed for the U.S. Air Force, is a general three-dimensional elliptic grid generation code based on the block structure. This code allows any number of blocks to be used to fill an arbitrary three-dimensional region. Any block can be linked to any other block (or to itself), with complete (or lesser) continuity across the block interfaces as specified by input. This code uses an elliptic generation system with automatic evaluation of control functions either directly from the initial algebraic grid and then smoothed, or by interpolation from the boundary point distributions. In the latter case, the arc length and curvature contributions to the control functions are evaluated and interpolated separately into the field from the appropriate boundaries. The control function at each point in the field is then formed by combining the interpolated components. This procedure allows very general regions, with widely varying boundary curvature, to be treated.

The control functions can also be determined automatically to provide orthogonality at boundaries with specified normal spacing. Here the iterative adjustments in the control functions are made by increments radiated from boundary points where orthogonality has not yet been attained. This allows the basic control function structure evalulated from the algebraic grid or from the boundary point distributions to be retained and thus relieves the iterative process from the need to establish this basic form of the control functions.

Alternatively, boundary orthogonality can be achieved through Neumann boundary conditions which allow the boundary points to move over a surface spline, the boundary point locations being located by Newton iteration on the spline to be at the foot of normals to the adjacent field points. Provision is also made for mirror-image reflective boundary conditions on symmetry planes.

Although written for 3D, the code can operate in a 2D mode on either a plane or curved surface. In the case of a curved surface, the surface is splined and the generation is done in terms of surface parametric coordinates.

The code includes an algebraic three-dimensional generation system based on transfinite interpolation (using either Lagrange or Hermite interpolation) for the generation of an initial solution to start the iterative solution of the elliptic generation system. This feature also allows the code to be run as an algebraic generation system if desired. The interpolation, though defaulted to complete transfinite interpolation from all boundaries, can be restricted by input to any combination of directions or lesser degrees of interpolation, and the form (Lagrange, Hermite, or incomplete Hermite) can be different in different directions or in different blocks. The blending functions can be linear or, more appropriately, based on interpolated arc length from the boundary point distributions.

The composite structure is such that completely general configurations can be treated, the arrangement of the sub-regions being specified by input, without modification of the code. The input is user-oriented and designed for brevity and easy recognition. For example, the establishment of correspondence, i.e., a branch cut, between two blocks requires only the simple input statement

```
$INPUT ITEM = "CUT", START = __,__,__, END = __,__,__, BLOCK = __,
ISTART = __,__,__, IEND = __,__,__, IBLOCK = __$
```

where START and END give the three indices of two opposite corners of the cut section on one block (BLOCK), while ISTART and IEND give the corners of the corresponding section on the other block (IBLOCK). The code sets up the point correspondence on the surrounding layers for complete continuity without additional input instructions.

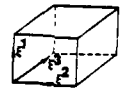
The features of this code and its use are discussed in Ref. 1. Detailed discussion of both the use and the operation of the code is given in Ref. 2, and some examples of applications have appeared in Ref. 3 and Ref. 4. The setup of general multi-blocked configurations is treated in Ref. 5.

The code is written in modular form so that components can be readily replaced, and an adaptive version of the elliptic generation subroutine has also been written.[6] The code is vectorized (CRAY-XMP) wherever practical and includes provision for separate storage of each block on the CRAY solid-state disk (or conventional disk) to allow very large grids to be generated.

CODE STRUCTURE

Composite Grid Structure

The grid is structured as follows: The entire three-dimensional physical region is filled with a set of interfacing hexahedrons, each of which corresponds to a rectangular computational block. Each of these computational blocks has its own set of right-handed curvilinear coordinates, ξ^i (i = 1,2,3): (independent of those in the other blocks):



Each block is identified by a number (starting with 1), and the size (the number of grid points in each curvilinear direction) of a block is set in the integer array

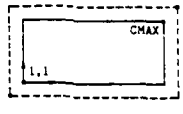
CMAX(i, block number) i = 1,2,3

The curvilinear coordinates of the grid points in the block thus assume the integer values

ξ^i = 1,2,...,CMAX(i,block number) i = 1,2,3

at the grid points in this computational block. The blocks do not have to be all of the same size, and the size of each is specified by input. It is also not necessary for an entire side of one block to correspond to an entire side of an adjacent block. It is only necessary that all of the corresponding blocks fit together to fill the physical region.

Each computational block is surrounded by an extra layer of points in order to allow connections across the interfaces in the physical region to be formed. All arrays that contain values for each grid point in a block are therefore dimensioned from 0 to one greater than the maximum number of points allowed in the block. Thus the surrounding layer of points outside the block corresponds to ξ^i = 0 on one side of the block and to ξ^i = CMAX(i, block number)+1 on the other:



(Actually, provision is made for still another surrounding layer of points, corresponding to ξ^i = -1 and ξ^i = CMAX+2, in order to provide connections for use in flow codes using two-point one-sided differences.)

Block Interfaces

The grid can be generated such that the grid lines cross the interface from one block to the next with complete continuity, with slope continuity, with only line continuity or discontinuously. With any degree of continuity, i.e., in all but the last case, adjacent blocks must, of course, have the same number of points on their common interface.

In the case of complete continuity, the interface is a branch cut, and the code establishes a correspondence across the interface using the surrounding layer

of points outside the blocks. This allows points on the interface to be treated just as all other points, so that there is no loss of continuity. The physical location of the interface is thus totally unspecified in this case, being determined by the code.

The case of slope continuity is accomplished simply by requiring the grid lines to intersect the interface orthogonally on both sides. This can be done either through Neumann boundary conditions, in which case point locations on the interface are determined by the code (with the shape of the interface specified by input), or by iterative adjustment of the control functions with the points on the interface specified by input.

Line continuity requires only that the same physical points be specified on the interface on each of the two blocks it joins, so that the points on the interface are completely specified by input. No continuity at the interface requires nothing at all, of course, and the adjacent blocks do not even have to have the same number of points on the interface in that case.

Sub-Block Structure

Blocks can be divided into sub-blocks for the purpose of generation of the algebraic grid and the control functions. Here point distributions on the sides of the sub-blocks can either be specified or generated by transfinite interpolation from the edges of the side. This allows additional control over the grid in general configurations and is particularly useful in cases where point distributions need to be specified in the interior of a block, or to prevent grid overlap highly curved regions.

Fundamental Arrays

In the following discussion the field arrays (which contain values at each grid point in a block), such as R given below, include the block number as a subscript. The code actually operates with data from only one block at a time in these arrays and hence this subscript is always unity in the code. The present explanation of usage is greatly simplified by the inclusion of the block number as a subscript, however.

The three Cartesian coordinates x_i $(i = 1,2,3)$ of the grid points in a block are in the real array

$$R(i, \text{block number}, \xi^1, \xi^2, \xi^3) \qquad i = 1,2,3$$

where (ξ^1, ξ^2, ξ^3) are the three curvilinear coordinates of the grid point in the computational block.

Each grid point in a block is given a classification set in the array

$$TYPE(\text{block number}, \xi^1, \xi^2, \xi^3)$$

This array, which is set up by the input, contains at each grid point one of the following alphanumeric values (the default is "FIELD" except on the surrounding layer where the default is "OUT")

TYPE = "FIX": indicates a point for which the Cartesian coordinates are not to be changed, e.g., a fixed point on a physical boundary.

TYPE = "FIELD": indicates a grid point for which the Cartesian coordinates are to be calculated by the grid generation system, e.g., a general interior point.

TYPE = "IMAGE": indicates an image point, i.e., a point on a block boundary or surrounding layer of points, for which the Cartesian coordinates will be kept equal to those at another (object) point in the same or another block.

TYPE = "REFLECT": indicates a point on the surrounding layer which is the mirror-image reflection in a plane physical boundary of a grid point just inside the boundary.

TYPE = "AVERAGE": indicates a special grid point on a block boundary which is the average of all the adjacent grid points.

TYPE = "NEUMANN": indicates a grid point on a boundary at which the grid lines are to be orthogonal to the boundary by the application of Neumann boundary conditions. (Such a point moves along the boundary.)

TYPE = "ORTHOG": indicates a grid point on a boundary at which grid lines are to be made orthogonal to the boundary by iterative adjustment of the control functions. (This leaves the boundary point fixed.)

TYPE = "OUT": indicates a point completely out of the computation, e.g., inside a body in the interior of a block.

The correspondence across the interfaces between the hexahedrons in the physical region is established in the integer array

IMAGE(__, block number, ξ^1, ξ^2, ξ^3)

where the first subscript assumes the values 0,1,2,3 as explained below. The Cartesian coordinates of points having TYPE = "IMAGE" (an image point) are kept equal to those of some other (object) point in the same or another block. The block number and curvilinear coordinates (ξ^i) of this object point are in the array IMAGE, where

		image point
block number		= IMAGE(0, block number, ξ^1,ξ^2,ξ^3)
object point	ξ^1	= IMAGE(1, block number, ξ^1,ξ^2,ξ^3)
	ξ^2	= IMAGE(2, block number, ξ^1,ξ^2,ξ^3)
	ξ^3	= IMAGE(3, block number, ξ^1,ξ^2,ξ^3)

Here the last four arguments of the array identify the image point, while the four values of the array identify the corresponding object point. This array is set up by the code. As an example of the function of the IMAGE array, if TYPE(IB,IC1,IC2,IC3) = "IMAGE", then the point with ξ^1=IC1, ξ^2=IC2, ξ^3=IC3 in block IB is an image point. The corresponding object point, say ξ^1=C1, ξ^2=C2, ξ^3=C3 in block B, is obtained from the IMAGE array as

```
B  = IMAGE(0,IB,IC1,IC2,IC3)
C1 = IMAGE(1,IB,IC1,IC2,IC3)
C2 = IMAGE(2,IB,IC1,IC2,IC3)
C3 = IMAGE(3,IB,IC1,IC2,IC3)
```

Then the Cartesian coordinates at the image point are set equal to those at the object point by

```
R(1,IB,IC1,IC2,IC3) = R(1,B,C1,C2,C3)
R(2,IB,IC1,IC2,IC3) = R(2,B,C1,C2,C3)
R(3,IB,IC1,IC2,IC3) = R(3,B,C1,C2,C3)
```

Block Storage

The code is set up to treat one block at a time, and hence the subroutines operate with only a single block in the field arrays. The blocks are stored either on disk files, one block to a file, or in the core storage arrays. The code keeps the number of the block presently in core and only accesses the storage when the next block to be treated is different from the last. All the field arrays are in one-dimensional form.

ALGEBRAIC GENERATION SYSTEM

Values of the Cartesian coordinates for grid points on any section of a block can be interpolated from already-specified values on the section boundary by transfinite interpolation[1]. This interpolation can be used to set points on boundaries for which the actual shape is not important, e.g. remote boundaries at 'infinity', or to set point distributions on interfaces between the blocks in the physical region for calculation of the control functions. This same type of interpolation is used by the code to generate an algebraic grid within each block, either as a final grid or to start the iteration for the elliptic system. In this case the section is the entire block.

Interpolation Type

The interpolation can be from the sides, edges, or corners of a section of block, corresponding to the portion of the section boundary to be matched by the transfinite interpolation. Cartesian coordinate values for all points on the section boundaries that are to be matched must have been set, of course. It is also possible to restrict the interpolation to less than the full dimensionality of the section.

The interpolation may be either Lagrange or Hermite, individually in each direction. For the Hermite case, the slope is made orthogonal to the boundary with a spacing determined either through specification, or through Lagrange transfinite interpolation from the point distribution on the section sides:

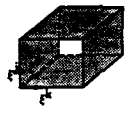
Finally, the blending functions for the interpolation can be linear or can be based on an interpolated arc length distribution constructed from the point distribution on the section boundary, also as in the above figure.

Interpolation Projectors

The transfinite interpolation is done by the appropriate combination of 1D projectors[7] for the type of interpolation specified. (Each projector is simply the 1D interpolation in the direction indicated.) For interpolation from all sides of the section, if all three directions are indicated and the section is a volume, this interpolation is from all six sides, and the combination of projectors is

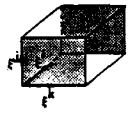
$$F_1 + F_2 + F_3 - F_1F_2 - F_2F_3 - P_3P_1 + F_1F_2F_3$$

while if only the two directions j and k are indicated, or if the section is a surface on which ξ^i is constant, the interpolation is from the four sides on which either ξ^j or ξ^k is constant



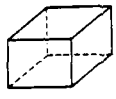
$$F_j + F_k - F_jF_k \qquad (i,j,k) \text{ cyclic}$$

With only a single direction i indicated, or if the section is a line on which ξ^i varies, the interpolation is between the two sides on which ξ^i is constant:



using only the single projector F_1.

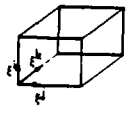
With interpolation from the edges of the section, with all three directions indicated and the section a volume, the interpolation is from all twelve edges:



using the combination

$$F_1F_2 + F_2F_3 + F_3F_1 - 2F_1F_2F_3$$

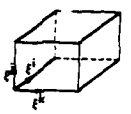
With only the two directions i and j indicated, the interpolation is from the eight edges on which either ξ^i or ξ^j vary:



with the combination

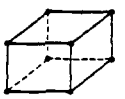
$$F_kF_\ell + F_{h\ 1} - F_1F_2F_3 \qquad \begin{matrix}(i,k,\ell)\\(j,m,n)\end{matrix} \text{ cyclic}$$

With only the single direction i indicated, the interpolation is from the four edges on which ξ^i varies:



using only F_jF_k, (i,j,k) cyclic.

Interpolation from the eight corners of the section



is done using $F_1F_2F_3$.

ELLIPTIC GENERATION SYSTEM

The code can function as either an elliptic generation system or an algebraic generation system. An algebraic grid is generated in any case to serve as a starting solution for the iterative solution of the elliptic system.

Elliptic System

The elliptic grid generation system[8,9] is

$$\sum_{m=1}^{3}\sum_{n=1}^{3} g^{mn}\, \underline{r}_{\xi^m\xi^n} + \sum_{n=1}^{3} g^{nn} P_n \underline{r}_{\xi^n} = 0 \qquad (1)$$

where the g^{mn} are the elements of the contravariant metric tensor:

$$g^{mn} = \nabla\xi^m \cdot \nabla\xi^n$$

These elements are more conveniently expressed in terms of the elements of the covariant metric tensor, g_{mn}:

$$g_{mn} = \underline{r}_{\xi^m} \cdot \underline{r}_{\xi^n}$$

which can be calculated directly. Thus

$$g^{mn} = (g_{ik}g_{j\ell} - g_{i\ell}g_{jk}) \,/\, g$$

(m,i,j) cyclic, (n,k,ℓ) cyclic

where g, the square of the Jacobian, is given by

$$g = \det |g_{mn}| = \underline{r}_{\xi^1} \cdot (\underline{r}_{\xi^2} \times \underline{r}_{\xi^3})$$

In these relations, $\underline{r}$ is the Cartesian position vector of a grid point ($\underline{r} = i\mathrm{x} + j\mathrm{y} + k\mathrm{z}$), and the ξ^i (i=1,2,3) are the three curvilinear coordinates. The P_n are the 'control functions' which serve to control the spacing and orientation of the grid lines in the field.

The first and second coordinate derivatives are normally calculated using second-order central differences. Provision is also made, however, for one-sided differences dependent on the sign of the control function P_n (backward for $P_n < 0$ and forward for $P_n > 0$). This feature is useful only to enhance convergence with very strong control functions. Provision is also made for skewed cross-derivatives, but this has been of little use.

The elliptic generation system is solved by point SOR iteration using a field of locally-optimum acceleration parameters.[10] These optimum parameters make the solution robust and capable of convergence with strong control functions.

Control Functions from Algebraic Grid

The three components of the elliptic grid generation system, Eq. (1), provide a set of three equations, that can be solved simultaneously at each point for the three control functions, P_n (n=1,2,3), with the derivatives here represented by central differences. This produces control functions which will reproduce the algebraic grid from the elliptic system solution in a single iteration, of course. Thus evaluation of the control functions in this manner would be of trivial interest except that in the code these control functions are smoothed before being used in the elliptic generation system. This smoothing is done by replacing the control function at each point with the average of the four neighbors in the two curvilinear directions (one in 2D) other than that of the function. Thus P_i is smoothed in the ξ^j and ξ^k directions, where i,j,k are cyclic. No smoothing is done in the direction of the function because to do so would smooth the spacing distribution.

The code generates an algebraic grid by transfinite interpolation from the boundary point distribution, as discussed above, to serve as the starting solution for the SOR iteration for the elliptic system. With the boundary point distribution set from the hyperbolic sine or tangent functions, which have been shown to give reduced truncation error[11-12], this algebraic grid has a good spacing distribution but may have slope breaks propagated from corners into the field. The use of smoothed control functions evaluated from the algebraic grid produces a smooth grid that retains essentially the spacing of the algebraic grid.

Control Functions from Boundary Point Distributions

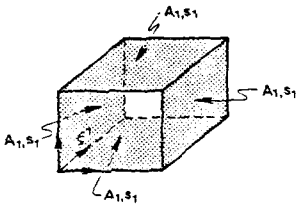
Control functions can be evaluated on the boundaries using the specified boundary point distribution in the generation system, with certain necessary assumptions (orthogonality at the boundary) to eliminate some terms, and then can be interpolated from the boundaries into the field. Earlier approaches[13] interpolated the entire control functions from the boundaries in this manner. More general regions can, however, be treated by interpolating elements of the control functions separately. (Some related work along these lines has appeared in Ref. 14).

The control functions on a line on which ξ^n varies can be expressed as

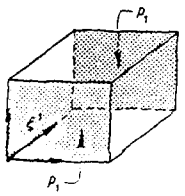
$$P_n = A_n + \frac{s_n}{\rho_n} \tag{2}$$

where A_n is the logarithmic derivative of the arc length, s_n is the arc length spacing, and ρ_n is the radius of curvature of the surface on which ξ^n is constant.

The arc length spacing, s_n, and the arc length contribution, A_n, to the control function are interpolated into the interior of the block from the four sides on which they are known by two-dimensional transfinite interpolation using linear blending functions:



The radius of curvature, ρ_n, is interpolated into the interior from the two sides on which it is known by one-dimensional interpolation using blending functions on the hyperbolic sine.



The control function is finally formed by adding the arc length spacing divided by the radius of curvature to the arc length contribution according to Eq. (2). (This procedure is discussed in more detail in Ref. 9.)

Iterative Adjustment of Control Functions

A second-order elliptic generation system allows either the point locations on the boundary or the coordinate line slope at the boundary to be specified, but not both. It is possible, however, to iteratively adjust the control functions in the generation system until not only a specified line slope but also the spacing of the first coordinate surface off the boundary is achieved, with the point locations on the boundary specified. In previous applications[9] the relations have been applied on the boundary, and the control function increments generated at the boundary have been interpolated into the field. In the present code, these relations are applied on each successive coordinate surface off the boundary, with the off-surface spacing determined by a hyperbolic sine distribution[11-12] from the spacing specified at the boundary. The control function increments are attenuated away from the boundary, and contributions are accumulated from all orthogonal boundary sections. Since the iterative adjustment of the control functions is a feedback loop, it is necessary to limit the acceleration parameters for stability. (More detail is given in Ref. 9.)

BOUNDARY CODE

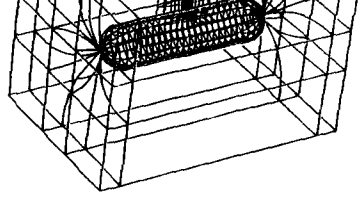
An auxilliary front-end code[16] has also been written to set up boundary data for input to the grid code. This auxilliary code builds boundary segments in response to a series of input commands which again are designed to be user-oriented, brief, and easily recognized. The following features are included:

(1). generation of generic plane conic-section or cubic curves.
(2). generation of cubic space curves.
(3). generation of generic conic-section surfaces.
(4). generation of cubic surfaces.
(5). generation of surfaces by stacking, rotating, or blending curves.
(6). extraction and concatenation of surface segments.
(7). transformation of surfaces by translation, rotation, and scaling.
(8). reversal or switching of point progressions on surface.
(9). establishment of point distributions by curvature and with specified end, or interior, spacings.
(10). establishment of surface parametric grids by transfinite interpolation.
(11). generation of tensor-product surfaces.
(12). generation of surfaces by transfinite interpolation.
(13). generation of grids on curved surfaces.

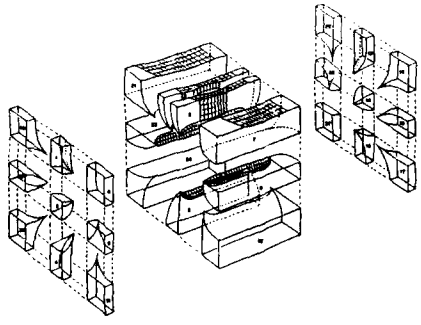
APPLICATIONS

In general the following details have been found to be advantageous. During the iteration, cuts on block sides are updated immediately after the block has been swept, since updating all of the cuts together after all of the blocks are swept can lead to oscillations near the cut. The SOR iteration is implemented in a symmetric manner, reversing the sweep direction after each iteration since this gives better symmetry, particularly with Neumann boundary conditions. The optimum acceleration parameters are essential to making the system robust. When the control functions are iteratively adjusted for boundary orthogonality, the use of one-sided, directed first derivatives is appropriate since the changes in the control functions can initially be quite large. Central differences are used in all other cases. The skewed cross derivatives, however, have shown little value. Finally, the evaluation of the control functions from the algebraic grid, followed by smoothing, has proved to be the most generally applicable approach, particularly in complicated configurations. Some examples appear below from Ref. 5:

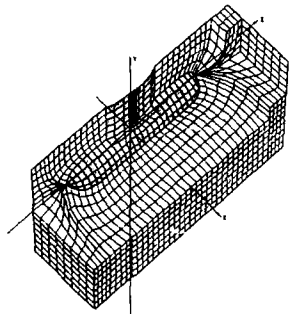
The following figure shows a 27-block structure for a pylon-store, constructed so as to transition from an O-type grid on the store to a rectangular macro-block that can be inserted into a C-grid about a wing.



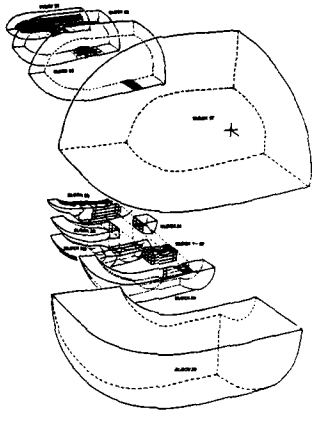
The next figure shows an exploded view of the physical region:



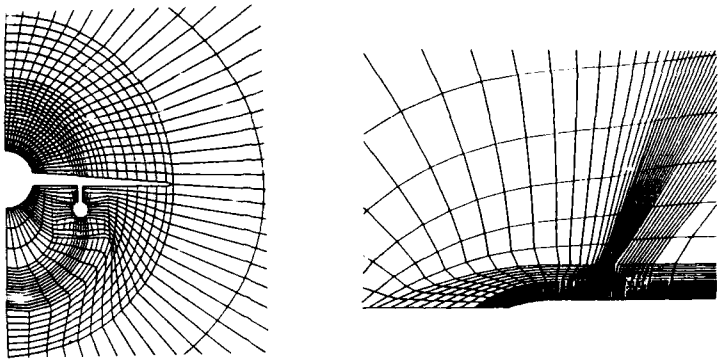
Finally, a cut-away section of the grid is shown:



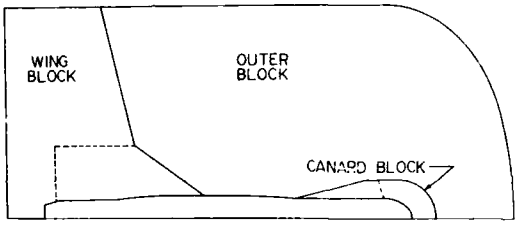
The insertion of this macro-block in the overall wing-body in the overall wing-body grid is indicated in the next figure:



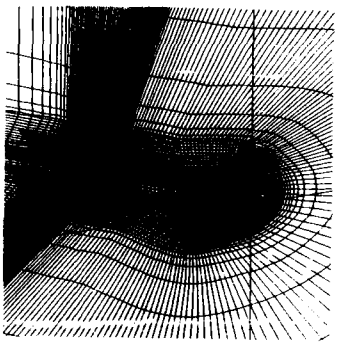
Finally, some surfaces from the resulting composite grid are shown:



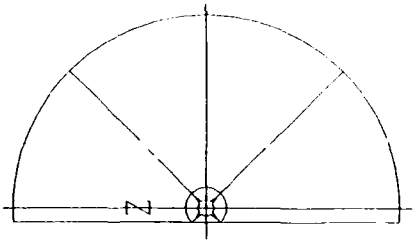
The following figure shows the block structure for a 12-block system about a wing-cornered store:



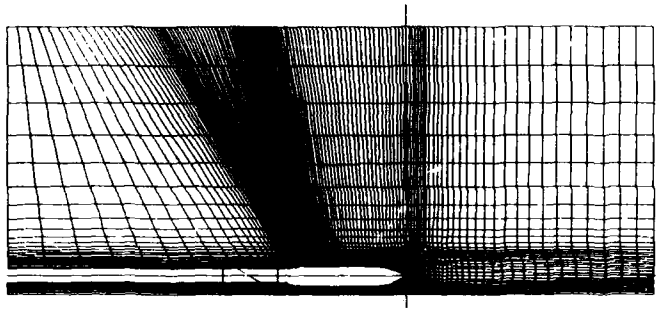
and a section of the grid follows next:



Finally, a 21-block structure for one of a pair of stores is shown (the bottom line is the symmetry plane):



followed by a section of the grid:



REFERENCES

1. THOMPSON, JOE F., "A Composite Grid Generation Code for General 3-D Regions", AIAA-87-0275, AIAA 25th Aerospace Sciences Meeting, Reno, Nevada, 1987.

2. THOMPSON, J. F., Project EAGLE Numerical Grid Generation System User's Manual, Vol. III: Grid Generation System, USAF Armament Laboratory Technical Report, AFATL-TR-87-15, Vol. III, Eglin AFB, 1987.

3. MOUNTS, J.S., MARTINEZ, A., and THOMPSON, J.F., "An Analysis of Elliptic Grid Generation Techniques Using an Implicit Euler Solver", AIAA-86-1766, AIAA Applied Aerodynamics Conference, San Diego, 1986.

4. BELK, D.M., and WHITFIELD, D.L., "3-D Euler Solutions on Blocked Grids Using an Implicit Two-Pass Algorithm", AIAA-87-0450, AIAA 25th Aerospace Science, Conference, Reno, 1987.

5. CHAE, YEON SEOK, "The Construction of Composite Grids for General Three-Dimensional Regions", Ph.D. Dissertation, Mississippi State University, August 1987.

6. KIM, HYUN JIN, "Three-Dimensional Adaptive Grid Generation on a Composite Structure", Ph.D. Dissertation, Mississippi State University, May 1987.

7. GORDON, WILLIAM J. and THIEL, LINDA C., "Transfinite Mappings and Their Application to Grid Generation", Numerical Grid Generation, Joe F. Thompson, (Ed.), North-Holland, 1982.

8. THOMPSON, J.F., WARSI, Z.U.A., and MASTIN, C.W., Numerical Grid Generation: Foundations and Applications, North-Holland, 1985.

9. THOMPSON, J.F., "A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure", to appear in Computer Methods in Applied Mechanics and Engineering.

10. EHRILICH, LOUIS W., "An Ad Hoc SOR Method", Journal of Computational Physics, Vol. 44, p. 31, 1981.

11. VINOKUR, MARCEL, "On One-Dimensional Stretching Functions for Finite-Difference Calculations", Journal of Computational Physics, Vol. 50, p. 215, 1983.

12. THOMPSON, J.F., and MASTIN, C. WAYNE, "Order of Difference Expressions on Curvilinear Coordinate Systems", Journal of Fluids Engineering, Vol. 107, p. 241, 1985.

13. THOMAS, P.D., "Composite Three-Dimensional Grids Generated by Elliptic Systems", AIAA Journal, Vol. 20, p. 1195, 1982.

14. THOMAS, P.D., "Stationary Interior Grids and Computation of Moving Interfaces", Advances in Computational Methods for Boundary and Interior Layers, J.J.H. Miller (ed.), Boole Press, Dublin, Ireland, 1984.

15. SORENSON, R.L., "Three-Dimensional Elliptic Grid Generation about Fighter Aircraft for Zonal Finite-Difference Computations", AIAA-86-0429, AIAA 24th Aerospace Sciences Meeting, Reno, 1986.

16. THOMPSON, J.F., Program EAGLE Numerical Grid Generation System User's Manual, Vol. II: Surface Generation System, USAF Armament Laboratory Technical Report, AFATL-TR-87-15, Vol. II, Eglin AFB, 1987.

4.10 ANALYTICAL SURFACES AND GRIDS

Helmut Sobieczky
DFVLR Institute f. Theoretical Fluid Mechanics
Göttingen, F. R. Germany

Summary

The use of analytical shape generation is decribed for wing-body configurations and flow boundary conditions. Flexibility in geometry definition allows for simple computational grid interpolation. A test case for experiment and code validation is illustrated.

Introduction

The use of computers has become essential for an efficient development of research tools in fluid dynamics. Large computers are needed to solve equations modelling fluid motion, smaller computers create graphic display of calculated physical phenomena. Flow field discretization is necessary for numerical solution methods: Model equations are solved within discretized portions of space surrounding the flow boundaries. Numerical flow solver techniques of various complexity - depending on the degree of simplifying the equations of motion - need computational grids of different properties defining resolution of space. Many grid types for flows past wings and bodies have been developed, the quality of flow solvers for practical applications is already measured by an ability to work with relatively simple grids formed around configurations of increasing complexity. Grid generation has therefore become a large part of the whole computational effort to model flows: equations and iteration techniques are used to find grid coordinates similarly as the subsequent effort needs to find the properties of the flow.

It has been found that the economics of grid generation is very much dependent of an ability to control surface metrics. Usually the surface is given by a more or less complete set of coordinates data providing supports for spline interpolation to obtain a surface grid. A much more precise definition of surfaces is possible if the shapes may - piecewise - be described by analytical relations. A modelling of a given configuration by analytical relations is of course tedious, but in design aerodynamics data generation with such methods is highly welcome because of the value of parametric studies.

We have developed a surface generator originally for wings designed for operation in transonic flow, stimulated by the sensitivity of the flow past given geometries to variations in transonic Mach number and lift. The design of aircraft primarily requires wing design concepts but we see that the fuselage and the wing-body junction influence the properties of a wing substantially.

It is intended here to show that, at least for the purpose of developing aerodynamic analysis codes and design concepts, surface generation is the most important part of grid generation and an analytical approach seems most useful for many applications, especially if workstations may be used for rapid interactive design and analysis. A strong connection to practical CAD/CAM and to experiment may also be established as will be illustrated here for a simple test wing configuration.

Development of a geometry generator

The beginnings of various users' geometry software were a necessity to define boundary conditions for their problem case studies. We recall the time when computational methods, e.g. for fluid mechanics, had to be tested with academic examples like the parabolic arc airfoil, the circle or sphere and ellipsoid, or similar. These examples are not too simple for practical flow studies, quite the contrary they include phenomena very difficult to model, but sometimes these phenomena are not relevant for practical cases, or they are not scaled properly compared to the topology of the flow past a more practical configuration. We - like others - had to solve therefore the problem how to define test cases for CFD as easy as defining a circle or an NACA 0012 airfoil but with intended local complexities to enforce occurrence of some aerodynamic phenomenon. The same approach should be of practical use to the designer in industry - at least there should be a straightforward way to extend software to practical tools. Presently we have a quite flexible family of codes able to generate a wide variety of wing-body configurations (Ref.1). To arrive there and continue toward complete aircraft we need a mathematical, an aerodynamic and an engineering background.

Mathematical relations

There are some very simple relations describing analytical functions connecting two given points in space. We may use algebraic and other analytical functions depending on additional quality requirements like tangents and curvature prescribed at the end points, or instead of smooth curvature some weakly singular behavior (Fig. 1). An exponential growth rate of the function may be of use as well as the simple parametric definition of parts of a circle by trigonometric relations. Another technique to generate discrete distributions, like grid points between given boundaries, makes use of a vector direction blending, based on the abovementioned functions used for distributions, clustering and connections. These explicit analytical basic tools are, of course, fast and simple which should be quite useful for large overall iteration loops, taking into account the whole design or analysis strategy.

Aerodynamic knowledge

Wings, bodies and their combinations for aerodynamic design, especially in the high speed regime may successfully use some geometry software package but for refined investigations - and these are what's needed by an already experienced community of design engineers - a flexibility to influence shapes, their gradients and their curvatures locally is essential: shape smoothness is necessary but not sufficient e.g. in transonic design where curvatures of wing upper surfaces need to be carefully balanced according to Mach waves of the flow field in two or three dimensions (Ref. 2). Computational grids should also be dense in regions of high or singular curvature, or where shocks are expected to occur.

Airfoil sections traditionally form a basic element of wing definition; our geometry code allows for given airfoil input data. We blow the sections up to give shapes with softened curvature peak at the leading edge, then we use a spline redistribution to uniformize point number and clustering of all sections serving as supports for a wing. The other possibility of a totally analytical geometry is to define airfoil generation by characteristic parameters, but their number might get too large for achieving desirable pressure distributions. Direct and inverse design methods, on the other hand, are available for transonic flow so that a resulting optimal 2D airfoil should be taken as a set of input data.

Less experience exists with optimal planforms and wing-body junctions so that our effort to generate these shapes analytically is intended to provide a multiplicity of variations for optimization strategies. A first application of combining this geometry generator with a fast transonic analysis code to find optimal wings is described in Ref. 3.

Engineering requirements and code practicability

The resulting shapes, though analytical and of arbitrary data density include realistic basic shapes with simple straight, uniformely rounded or other elements which allow a comparison with known case studies as well as they include simplifications dictated by engineering constraints.

Input for the generator code has been developed to control a selection of function parameters: besides coordinates these parameters include tangents and curvature or singularity exponents, controlled by a curve key and function identifier. The key identifies the parameters supporting a special curve like a leading edge shape or a body crown line, the function identifier selects a certain function formula to model a portion of the special curve. The resulting set of data for all definition curves in 3D space is useful for interactive work on a graphic work station: Axonometric or perspectivic views and selection of grid portions allows for high productivity because of the extremely fast explicit computation.

For wind tunnel model production the surface normals are used with a given tool radius to define the cutter path for NC milling (Fig. 2). Surface undercuts at concave portions are monitored, so the maximum tool radius for smoothest surfaces at the different milling steps is found.

Surface generation

Some elements combined to form a configuration are of prime importance and so we focused our efforts to generate fuselages, wings, flow wake sheets and flow field

boundaries including internal flow in nozzles and diffusors. A generalization of superelliptic quarter arcs gives smooth cross sections, with curvature singular, finite or vanishing, depending on the exponents used. Restrictions to shape complexity led to adding an ability to define basically rectangular cross sections with rounded corners. These cross sections - defined by half axes and corner radii - are thread onto a curve in space, given by x,y,z as functions of its arc length. Function subroutines provide first and second derivatives, so the normal plane to a curve in 3D space can easily be given. These shapes are applied to generate fuselages and channels (Fig.3).

Coordinates of a body surface grid may be defined now in various ways but most useful sceems an explicit definition of the spanwise coordinate as function of streamwise and vertical coordinate: we use this for smoothly fix the wing root onto the body.

Wing parameters require a definition of selected spanwise section stations, leading and trailing edge shape, dihedral, twist axis, twist distribution, airfoil thickness variation and parameters to blend-interpolate the given support airfoils between their spanwise locations. We presently treat the fairing of a wing root, or the fillet, like airfoil sections. An isolated wing with a fillet opens therefore like a trumpet at its root (Fig. 4). Wing sections are thread onto a 3D twist axis allowing for applying section angle of attack and a vertical bending of the wing. The wing root area may now be projected toward the body so that no gaps are left between body and wing. If wing fillets are provided we may form a completely smooth wing-body junction.

Many present flow analysis computer codes require a suitable choice of a computational wake sheet, possibly adjusted to the flow wake iteratively. We have therefore provided parameters to continue the wing sections beyond the trailing edge, downstream toward an exit plane. Vortex roll up at the tip and near the body may be modeled. Finite trailing edge thickness results in two parallel sheets suitable for inserting a fine additional grid block essential to model viscous flow from a blunt base downstream. The wake of the root section is projected toward the body surface, allowing to interpolate a simple C-type body surface grid between upper and lower crown lines and the wing root section plus wake. Body and wing have now one type surface metrics with sections from the plane of symmetry to the wing tip (Fig. 5).

Similar to wake sheets we treat far field boundaries like body geometries. Among the many possible grid topologies we had priorities for CO-type 3D grids for our analysis codes. So we generate a body with round nose and C-type spanwise sections. CO grids allow for a refined wing tip analysis, the tip far field C section is reduced to a cut in the rounded side. CH grids need a wing tip extension and an open far field. Both types of grids are generated automatically by the grid interpolation routine (Fig. 6).

Grid interpolation

In this paper we stress the importance to achieve a maximum flexibility in generating surface grids (including wake and far field) as the best prerequisite to define mesh distributions in space. With bounding surfaces given, an interpolation may be carried out in many ways depending on configuration complexity. Many authors use partial differential equations. We have provided surface grids to serve as boundary conditions for elliptic grid generators, e.g. to obtain a grid in a channel. Here we restrict a description of our experience to grid interpolation by analytical methods, added as a subprogram to the surface generator.

A simple vector combination technique is used in this code, it requires start and end point in space, a starting direction and a clustering function for the interpolated points. The problem of intersecting grid lines is relatively easily controlled and avoided for the configurations studied. A recent addition confirms this easy control: We place the "far field" boundary relatively close to the body, with sections and points distributed so that surface normals pass end points of the interpolated grid close enough to avoid too strong turns and intersections. This boundary has turned to a near- or midfield boundary; from there starting with end point directions, grid trajectories may continue to a new real far field boundary, far away in free stream and with coarse mesh near this outer surface (Fig. 7).

This continuation may also be used to change from a C- to an H-type grid for flow analysis codes handling block-structured grids. An application is the grid around a wing or wing-body combination in a rectangular channel, where the near-field C type boundary is surrounded by an H type block conforming with the channel walls and also allowing a grid clustering at the walls for modelling viscous flow or ventilation at wind tunnel walls.

An Application: DFVLR-F5 Configuration

In our effort to improve numerical methods in CFD, precisely defined test cases are needed. The geometry generator developed here offers many possibilities to provide configurations for such purpose. A first example was chosen carefully between two extremes: Creating the model of only a "bump" body forming some 3D displacement in the flow certainly already allows for complicated viscous flow phenomena experiments, their physical interpretation and computational verification. Designing a realistic wing-body configuration with a supercritical lifting wing, on the other hand, is already possible with this generator using present aerodynamic experience. The goal was the compromise of trying to define a "clean experiment", avoiding uncertainties of wind tunnel corrections but still obtaining data related to typical measurements on swept wing configurations for transport aircraft. Using half model technology in a 1x1 m transonic tunnel allowed for a larger wing but required a careful flow control at the splitter plate leading edge, to avoid the thick boundary layer on one of the tunnel walls. The slotted walls were completely closed, just suction in the splitter plate bypass channel was provided - among other devices - for controlling the plate leading edge flow (Fig. 8).

The configuration presented as a test case is a non-lifting wing with pronounced fairing on the splitter plate wall. The surface generator provided data for NC milling of the model. Geometry accuracy achieved by this approach was remarkably high, confirming the possibility to use the code for model production.

Airfoil design and wing geometry definition, model production and the wind tunnel experiment were the first part of the DFVLR-F5 project. The second part is a data evaluation and offering geometry and flow boundary conditions from the experiment (Ref. 4) to interested partners in the CFD community, followed by a workshop to compare computational results. The test case lends itself to the development of various computational analysis codes, these will use grids with different topology. Measured flow data (pressure, temperature and velocity components) were modeled analytically to a reasonable accuracy defining flow quantities on any chosen computational grid in inlet and exit planes (Fig. 9). Accepting these boundary conditions as good models for measured values, we have completed a precise geometry input by an equally precise flow boundary.

Our own efforts to improve potential, Euler and boundary layer codes as well as develop new solvers for the Reynolds averaged Navier Stokes equations add to experience how to choose grid topology, density and clustering. Potential flow results give a first insight into flow quality at wing root and tip, N/S analysis of 2D airfoil flow past the swept wing section gives information about required grid quality subsequently applied to a 3D version of the N/S code in free stream and in channel flow (Fig. 10). The goal is finally to learn about a most economic use of all codes in global and zonal approaches: all of them require rapid and flexible handling of geometrical problems.

Further use of this wing is, in combination with a generic body, the development of design and optimization strategies: studying the reaction of flow quality to the changes in geometry by a systematic variation of certain parameters leads to a better understanding of flow sensitivity and consequently to better tools for design aerodynamics.

Concluding remarks

The use of analytical geometry and grid generation was illustrated by definition of wing-fuselage and other configurations. Flexibility in shape definition and surface metrics generation includes a large part of the work necessary for obtaining acceptable computational grids. The fast solution of explicit analytical relations invites to the interactive design of geometries and grids on a graphic work station. A generated example was used to precisely define a transonic flow experiment for analysis codes development.

References

[1] Sobieczky, H., Geometry Generation for Transonic Design
Recent Advances in Numerical Methods in Fluids, Vol. 4, Ed. W.G. Habashi, Swansea: Pineridge Press 1985, pp. 163 - 182

[2] Sobieczky, H.; Seebass, A.R., Supercritical airfoil and wing design, Ann. Rev. Fluid Mech. 1984, 16, pp. 337-363

[3] Cosentino, G. B., Holst, T. L., Numerical Optimization Design of Advanced Transonic Wing Configurations
J. Aircraft Vol. 23, 1986, pp. 192 - 199

[4] Sobieczky, H., DFVLR-F5 Configuration for Computational and Experimental Aerodynamics
DFVLR Report to appear 1987

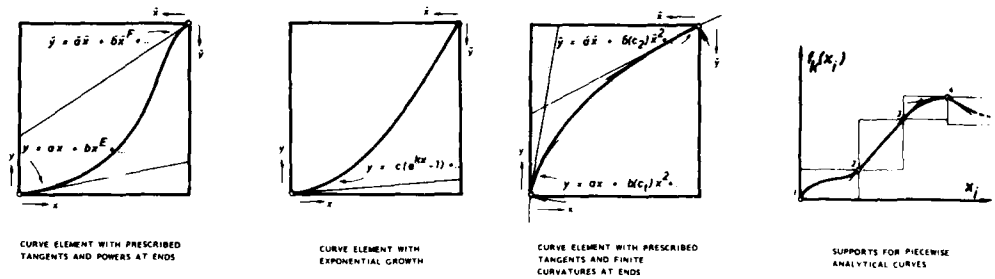


FIGURE 1. ANALYTICAL RELATIONS FOR THE DEFINITION OF SHAPE ELEMENTS, DISTRIBUTION AND BLENDING FUNCTIONS

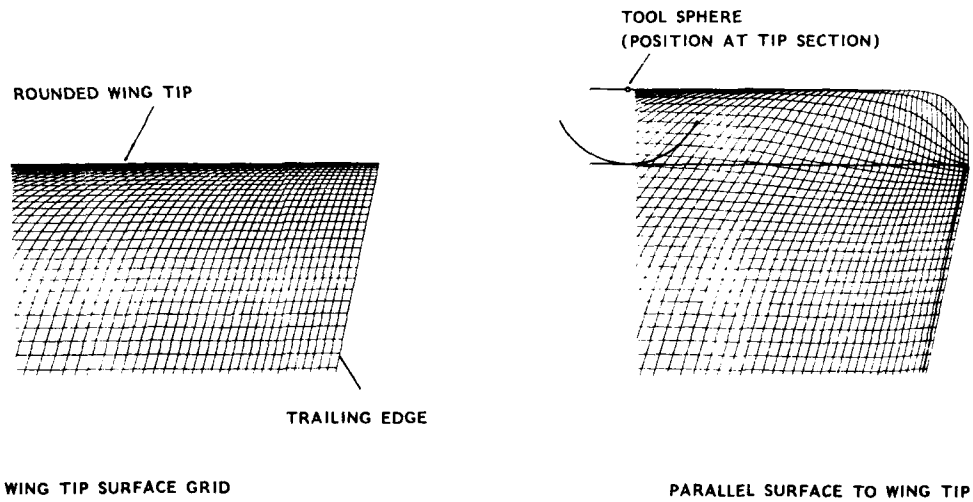
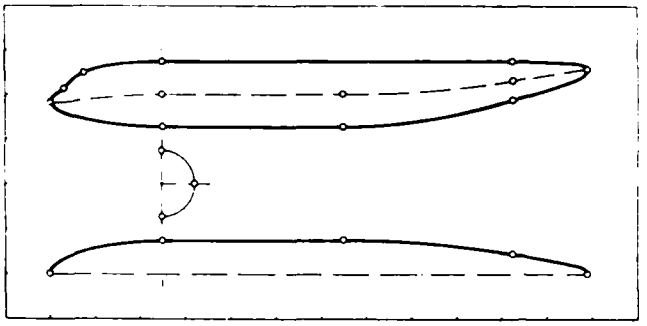
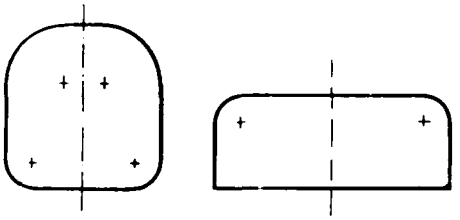


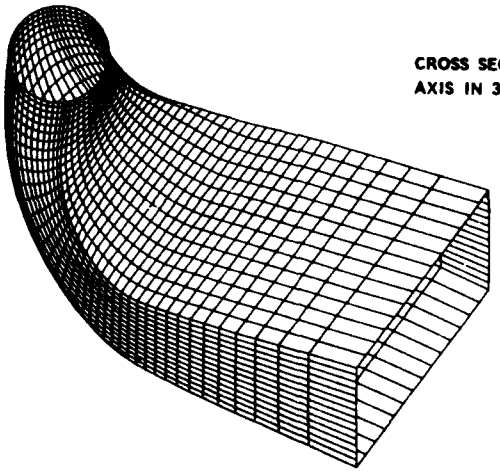
FIGURE 2. CUTTER PATH GRID FOR NC MILLING
(TOP VIEW OF WING TIP AREA NEAR TRAILING EDGE)



BODY SUPPORT CURVES

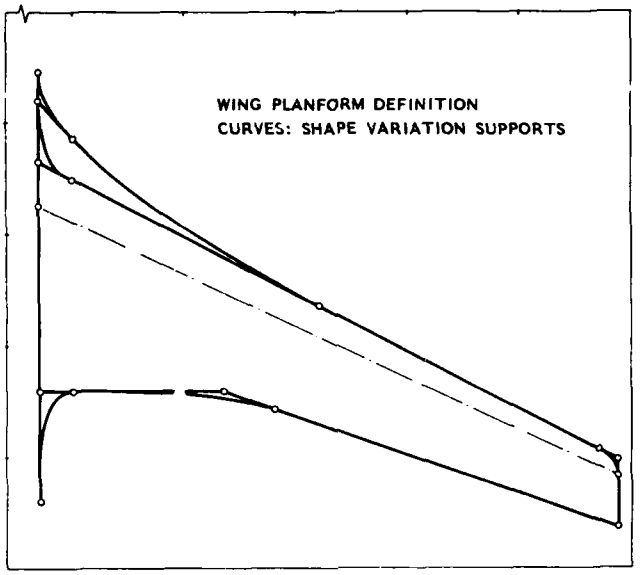


CROSS SECTION PARAMETERS FOR BODIES AND CHANNELS



CROSS SECTIONS THREAD ONTO
AXIS IN 3D SPACE

FIGURE 3. BODY OR CHANNEL SUPPORT CURVES, CROSS SECTION PARAMETERS, CHANNEL GEOMETRY EXAMPLE: TURBINE DRAFT TUBE



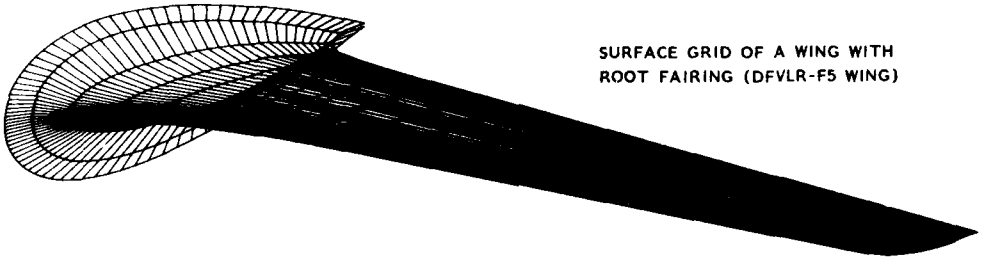


FIGURE 4. WING GEOMETRY SUPPORT CURVES AND SURFACE GRID

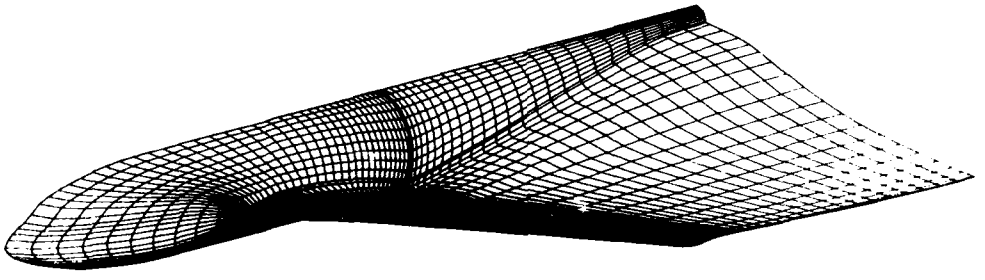


FIGURE 5. WING - BODY CONFIGURATION MODEL WITH COMPUTATIONAL WAKE SHEET

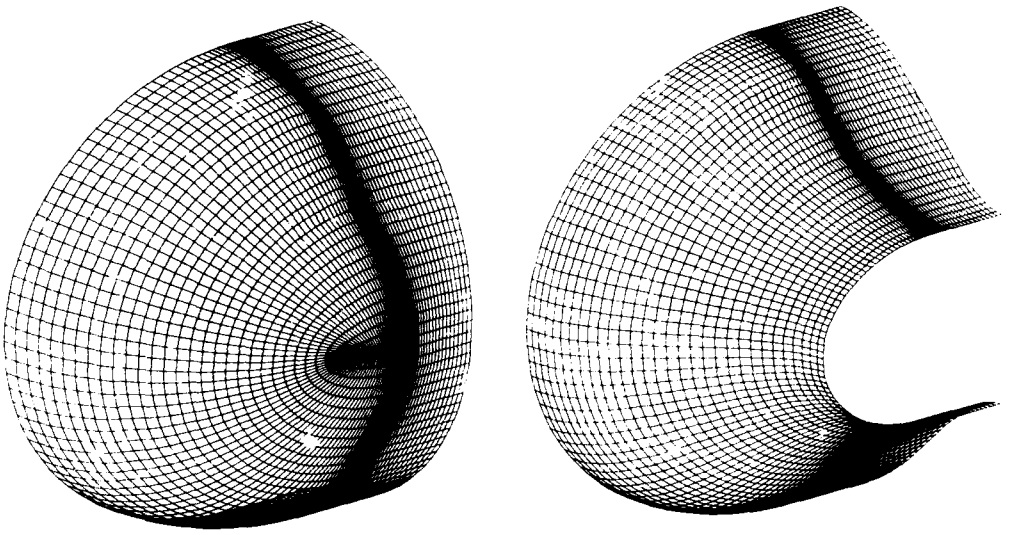


FIGURE 6. FAR FIELD SURFACES FOR CO- AND CH-GRIDS

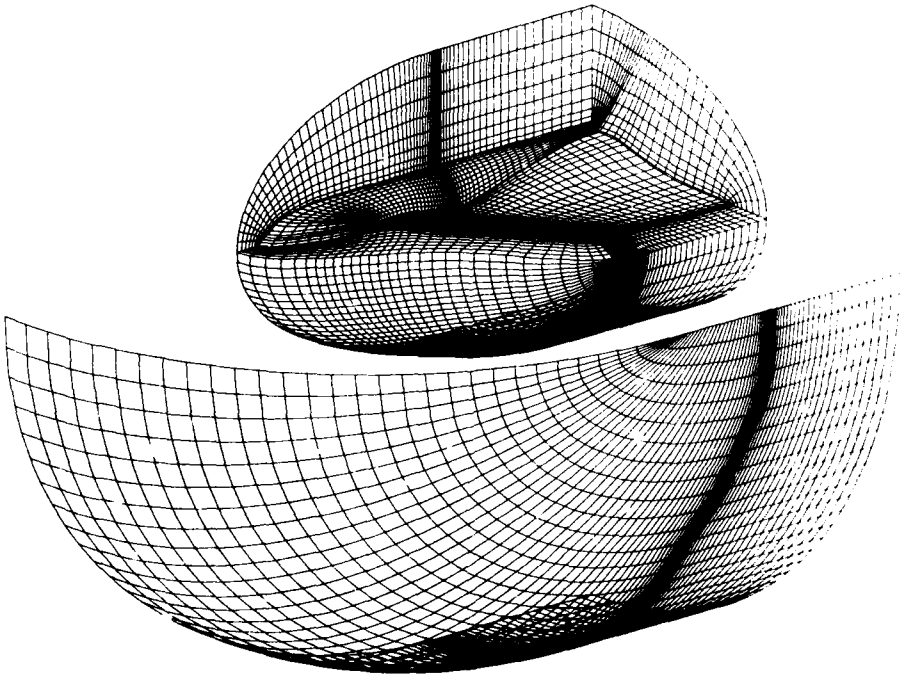


FIGURE 7. WING - BODY CONFIGURATION WITH CO TYPE GRID,
GRID CONTROL BY PRESCRIBED MID-FIELD SURFACE

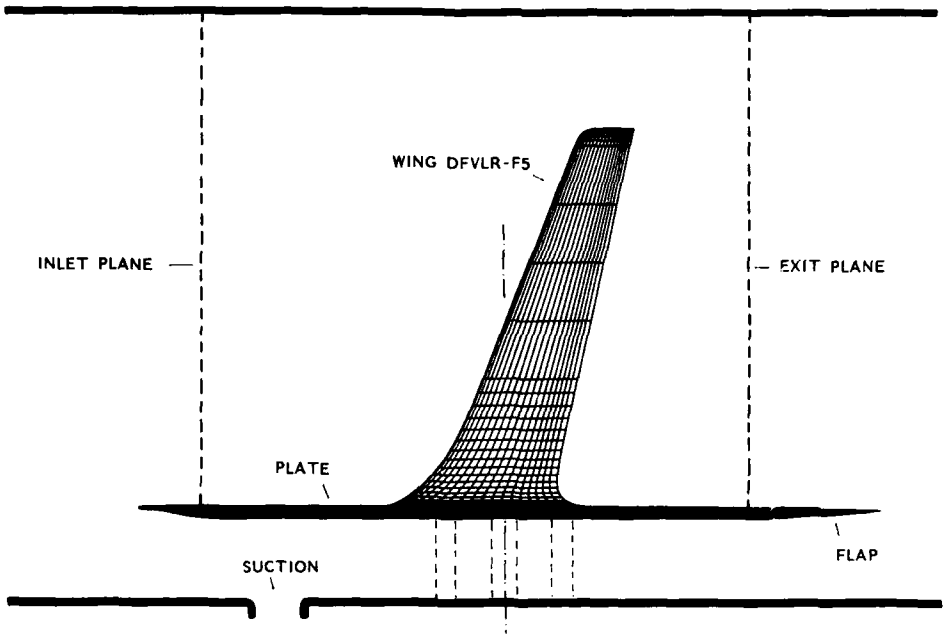
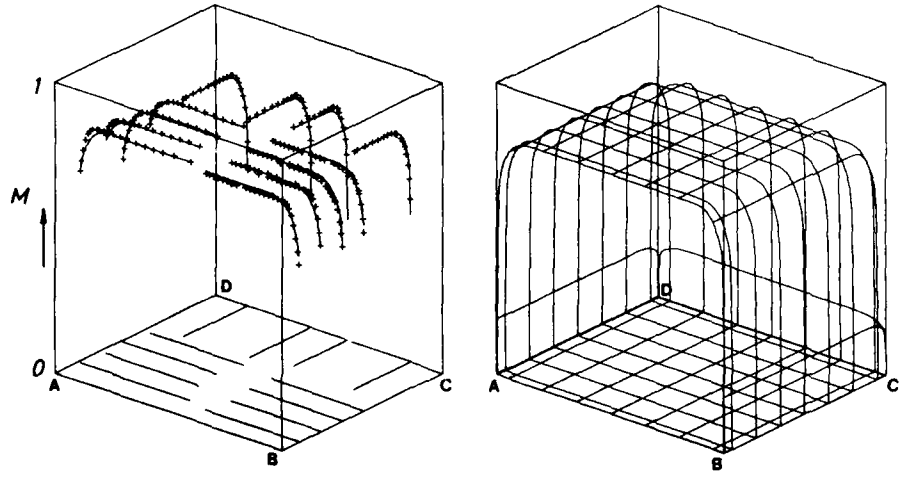


FIGURE 8. DFVLR-F5 WING - PLATE CONFIGURATION IN TRANSONIC WIND TUNNEL: FLOW FIELD MEASUREMENTS AT INLET AND EXIT PLANE

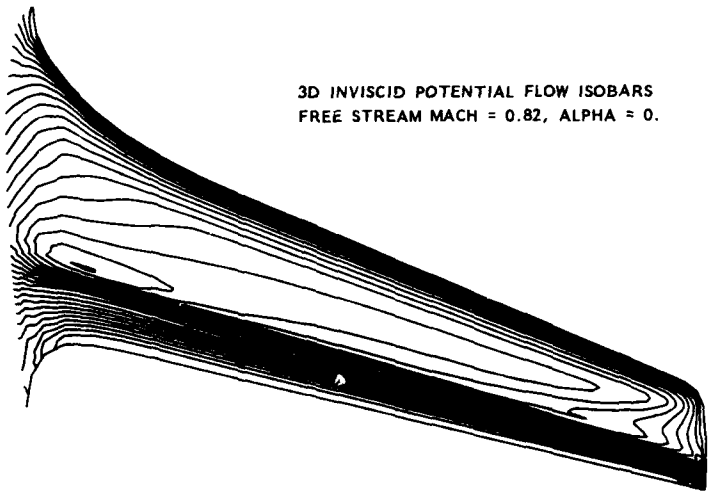


(AB: PLATE, BCDA: WIND TUNNEL WALLS)

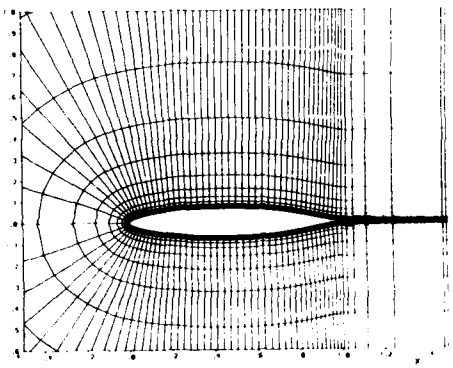
COMPARISON OF ANALYTICAL MODEL WITH MEASUREMENTS ON PROBE PATHS

ANALYTICAL MODEL VELOCITY PROFILE ON INLET PLANE COMPUTATIONAL GRID

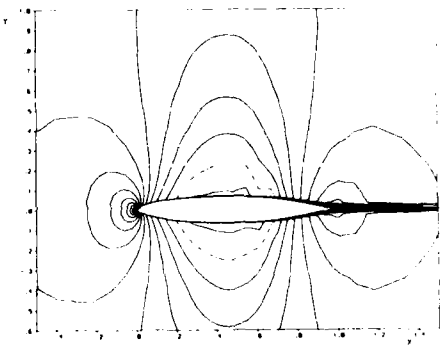
FIGURE 9. MODELING INLET PLANE FLOW PARAMETERS FROM EXPERIMENTAL DATA FOR COMPUTATIONAL BOUNDARY CONDITIONS DEFINITION



3D INVISCID POTENTIAL FLOW ISOBARS
FREE STREAM MACH = 0.82, ALPHA = 0.



2D GRID FOR WING SECTION



2D NAVIER/STOKES RESULTS
(REISTER/SCHWAMBORN CODE)
FREE STREAM MACH = 0.78, RE = 2 MILL.,
ALPHA = 0.

3D GRID WITH EXPONENTIAL CLUSTERING
NEAR SURFACE FOR N/S CODES
(WING TIP)

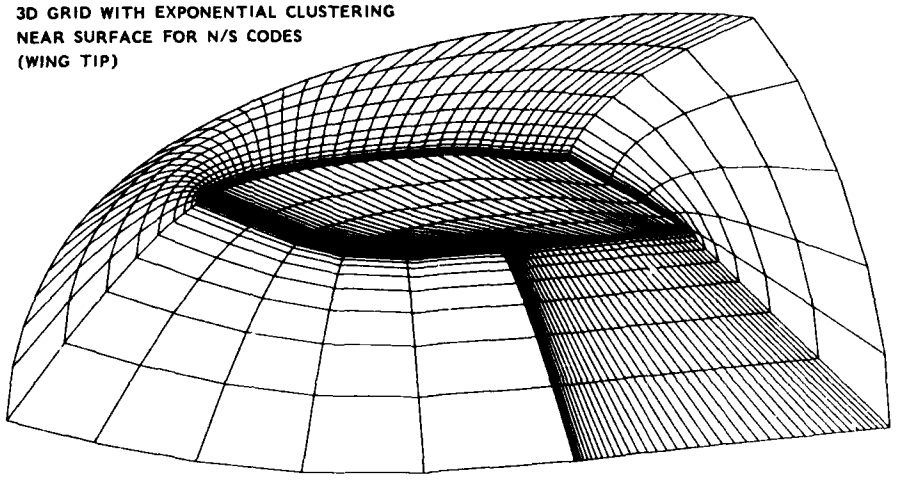


FIGURE 10. GRID DEVELOPMENT AND COMPUTATIONAL RESULTS FOR
DFVLR-F5 CONFIGURATION

4.11

Mesh Generation for Industrial Application of Euler and Navier Stokes Solvers.

W. Fritz, W. Haase, W. Seibert
Dornier GmbH, Theoretical Aerodynamics, Friedrichshafen, F. R. Germany

1. Introduction

In recent years, there has been a considerable increase in the ability to compute flow fields about three-dimensional configurations. The level of the field equations which could be considered has increased from the small disturbance potential methods in the early seventies over full potential and Euler methods to Navier-Stokes methods. The complexity of the geometry which could be considered has also increased from isolated wings over wing-fuselage representations to wing-body-tail geometries and more nearly to complete aircraft geometries. Because of the great generality of the most commonly used finite volume technique the flow field around any configuration can be solved if it is possible to map the configuration and the surrounding field into the rectangular computational space. This mapping is done by the grid generation in the physical space.

Already in 1974 Thompson, Thames and Mastin [1] described a method whereby a grid could be generated around an arbitrary two-dimensional body. This technique which involves the solution of non-linear elliptic partial differential equations for the grid points, readily generalises to three dimensions, and in principle, provides the means of grid generation for complex shapes. Meanwhile this technique is well known as the "Standard Thompson Approach" and is the basis of most of the grid generation techniques. In the past ten years or so, grid generation has been of secondary importance, but the fact remains, that for the flow solvers to reach their full potential, robust grid generation techniques for complicated aerodynamic configurations must be developed.

To this end we present 3 different methods which can be characterized as automatic grid generation for complete aircraft configurations, completely interactive grid generation and generation of solution adaptive grids for Navier Stokes calculations.

2. Block Structured Grid Generation around Complete Aircraft Configurations

The block structured grid generation technique as it is given for example in the references [2], [3] or [4] divides the computational domain into multiple rectangular blocks, which can be defined arbitrarily to produce surface-fitted grids whose structure follows the natural lines of the configuration. The Figure 1 shows in principle such a block structuring of the grid around an aircraft. Such a subdivision of the physical space, when properly carried out, can adapt to complex configurations with multi-components in such a way as to reduce grid skewness near the boundaries and provide good grid behaviour around the surface slope discontinuities. It undergoes also the storage restrictions of existing computers for fine 3-D grids because during the grid generation as well as during the flow solution, only part of the complete 3-D field has must be present in the main storage

The typical block-structured grid generation process can be described as follows:

- Definition of the overall block structure according to the natural lines of the configuration. (Definition of the block corner points).
- One-dimensional block perimeter discretization. (Connection of the block corner points). This can be done in the simpliest way, connecting the block corner points by straight lines and taking an evenly spaced point distribution along those lines. But such a perimeter discretization can produce large discontinuities in the spacing and in the slope of the coordinate lines accross the block boundaries. The presented method makes large efforts to get smooth perimeter lines accross the block boundaries.
- Two-dimensional grid generation for each block surface. Such block surfaces can be pysical surfaces (fuslage, wing etc.), free surfaces and far field surfaces which are bounded by the block perimeter lines. This block surface discretization can be done either by algebraic interpolation or by the solution of an elliptical PDE.
- Three-dimensional grid generation for each block. A block here is a subdomain of the 3-D grid, bounded by the block surfaces. This also can be done in 3 levels:

  -Algebraic interpolation.
  -Section wise solution of a 2-D PDE.
  -Solution of a 3-D PDE

  where each level can be the initial solution for the next level.

The partial differential equations which are solved for 2-D and 3-D grid optimization are derived from the Poisson equation of the form:

$$\xi_{xx} + \xi_{yy} + \xi_{zz} = P(\xi,\eta,\zeta)$$

$$\eta_{xx} + \eta_{yy} + \eta_{zz} = Q(\xi,\eta,\zeta)$$

$$\zeta_{xx} + \zeta_{yy} + \zeta_{zz} = R(\xi,\eta,\zeta)$$

where (ξ,η,ζ) are the computational, and (x,y,z) the physical coordinates. P, Q and R are source terms which control the interior grid spacing. The above equations can be transformed to the computational coordinates (ξ,η,ζ) by interchanging the role of dependent and undependent variables. This leads to a quasi-linear elliptic system of equations:

$$A X_{\xi\xi} + B X_{\eta\eta} + C X_{\zeta\zeta} + D X_{\xi} + E X_{\eta} + F X_{\zeta} = 0 \qquad (1)$$

wherein the X = (x,y,z) are the cartesian coordinates of the grid points. These equations are solved for each block by succesive line over relaxation (SLOR). The coefficients A to F are constant or specified functions used for grid control. The grid control terms are defined along each block boundary and then interpolated across the interior grid. At the boundaries, the values are estimated by the condition, that all derivatives normal to the boundary in equation (1) vanish. It is also possible, to modify the control functions in an interactive way.

The method generates a block structured grid of the H-type and uses a coordinate system as follows: The X-coordinate direction is the centerline of the fuselage with the positive direction running from the nose to the tail. The Z-coordinate direction is in the spanwise direction (left wing), while the Y-coordinate points upward from the fuselage centerline.

One main difference to other existing methods is the fact, that the grid is not divided into blocks at the beginning of the grid generation process with following grid generation for each block. As far as possible, the subdomains are kept as large as possible during the grid generation. So the complete grid generation is splitted into 3 separate parts: Surface grid generation, block surface discretization and volume grid generation, where the surface grid generation and the block surface grid generation are performed independently of the final block structure. By doing this, it is possible to get grids with very smooth gridlines across the block boundaries. Only at the end of the grid generation process, after the volume grid generation, the grid is divided into the final block structure.

The first step in the generation of the grid is the input and the preparation of the configuration geometry. Figure 2 shows a typical geometry definition. Fuselage wings and tail are defined by some definition sections. As the fuselage will be mapped into a hexadron in the index space, the perimeter lines have to be found in the physical space. This is done automatically (Figure 2), but the block perimeter lines can be corrected by the user. Wing fuselage intersections can be included in the geometry definition, but it is not mandatory.

The next step is the generation of the surface grids. First the surface grids for the wing, the canard and the vertical tail are generated separately using a square root coordinate transformation. The fuselage surface grid is built up starting at the different wing-fuselage intersections. If the wing-fuselage intersections are not included in the geometry definition, they are caluclated projecting the inboard wing sections on the fuselage surface. After the discretization of the block perimeter lines the surface grid points are optimized by the solution of a 2-D version of the PDE (1) on the fuselage surface. This optimization can be done interactively, thereby modifying the source terms in equation (1) for the grid spacing. Furthermore it is possible to select arbitrarily bounded regions in which the surface grid can be smoothed by the solution of the PDE (1) So it is possible, to get smooth grid lines across the block boundaries by selecting regions which overlap the block boundaries. To do this, the fuslage is transformed into a coordinate system with the coordinate x in streamwise direction, a coordinate θ in circumferential direction which can be either the arclength or the local angle, and a coordinate r in radial direction. Now the distribution of the x- and θ-coordinates is optimized by the solution of the 2-D PDE. The radial coordinates of each surface grid point are obtained by a bi-cubic spline approximation of the input geometry.

During this surface grid generation, only the surface grids of each component are stored separately as two dimensional arrays (x(i,j), y(i,j) and z(i,j), where i and j are the two characteristic computational surface coordinates).

Figures 3 and 4 show two typical surface grids.

The third step is the transfer of the surface grids into the 3-dimensional index space and the definition of all block boundary points in the far field planes, the discretization of those far field planes, which is very simple, as in the far field planes very regular grids (straight, paralell lines) are used in those planes. If this has been done, the grid is stored sectionwise from inboard to outboard on an external dataset as it is indicated in Figure 5 for the index space. (In the index space, the index j runs in streamwise direction starting at the upstream far field, the index j runs from bottom to top and k from inboard to outboard). In each k-section, the point distribution along the outer boundaries and along the the surface grid lines is known, all the other coordinates are still unknown.

Next is the discretization of the internal block surfaces. This is done in 3 steps:

- Discretization of the block surfaces k = constant.
- Discretization of the block surfaces i = constant.
- Discretization of the block surfaces j = constant.

As it is shown in Figures (6) (7) and (8). In each of the above steps, only one block surface is required during the discretization. Again each block surface k = const., i = const., or j = const. is stored as two dimensional arrays X(i,j) for block surfaces k = const., X(k,j) for block surfaces i = const. and X(k,i) for block surfaces j = const. Each block surface is updated as follows: First the perimeter lines which divide each block surface into sub-blocks are estimated. These lines are taken as cubic parabolas with specified slopes at the end points. For the one-dimensional discretization along these block perimeter lines arithmetric or geometric or user specified stretching functions are used as weighting functions. Then the grid of each subdomain is generated either by algebraic interpolation or by solution of the 2-D version of the PDE (1). To get smooth grid lines, those subdomains are chosen as large as possible. For example the block surface grid in Figure 9 is divided into the following 4 subdomaines:

- The complete region between fuselage upper side and the upper far field from the upstream far field boundary to the downstream far field boundary.
- The complete region between fuselage lower side and the lower far field from the upstream far field boundary to the downstream far field boundary.
- The region between upstrem far field and the beginning of the fuselage from the lower to the upper far field.
- The region between end of the fuselage and the downstrem far field from the lower to the upper far field.

For each of those subdomaines the PDE (1) can be solved. As the subdomaines are overlapping across the block boundaries, the block boundaries are also smoothed by the solution of the PDE (1). Finally the block surface grids can be optimized in an interactive way by applying equation (1) for aritrarily defined subdomaines and modifying the source terms for equation (1). Due to this block surface grid generation, all the block suface grids have

- continuity of coordinates
- continuity of slopes
- and as far as possible continuity of cell size

across the block boundaries.

Figures 9 and 10 show two of such surface grids.

As the surface grids for each sub-region now being generated, the volume grids can be generated for each sub-block separately either by algebraic interpolation, sectionwise solution of a 2-D PDE or by the solution of the 3-D PDE for each block. Similar to the block surface grid generation the subdomains for the volume grid generation, for which the grids are generated separately, are defined as large as possible. Therefore the volume grids are very smooth across the block boundaries. Figure 11 gives an impression of the 3-D grid arrangement.

The final grid is arranged blockwise with uniform boundary conditions for each block, so that it can be used by a block structured flow solver. Details of an Euler solution in the above grid are given in ref [5]. It can be seen, that the block structure allows the generation of fine grids for realistic aircraft configurations which seems to be important for future Navier-Stokes calculations. It furter avoids the storage restrictions of existing computers for fine 3-D grids. Because of the small main storage requirements of the surface and block surface discretization (less than 1 MB on the IBM OS/MVS system) this part of the method can run interactively and therefore allow interactive control and optimization by the user. It is then possible to generate grids with a total number of 4.9 million grid points with this version.

3. Graphic Interactive Grid Generation

While observing the development of hard- and software within the CAD/CAM area or in computer graphics generally, the decision arises almost mandatorily to rearrange the entire geometric preprocessing to a graphic-interactive solution.

The advantages are obvious: within a dialog and under permanent visual control step by step (and even backwards) a basic geometry can be upgraded to a final network within one session. Errors can be cancelled immediately since they are easy to recognize and possible variations can be tried at smallest expenditure of time. In addition, the comprehensive possibilities of 3D-representation of modern workstations with local intelligence, today with smallest CPU-usage, supply picture sequences, which some years ago were only possible with complex trick film techniques.

With this basic objectives the approach described in the following is only one of several possible ways, however the principles of a necessary new concept are explained. The entire procedure of the grid generation, independently of the supplementary aids with which it is accomplished, can be divided into two sub-tasks: geometry-preparation and grid-generation. The first part leads to a configuration description by means of suitable geometrical elements, the second contains the algorithms which are necessary for the discretization.

Since for pure geometrical tasks several interactive program systems exist already within the so called Computer-Aided-Design area, for the solution of the first task, existing software and an appropriate installation should be used. For the execution of the actual grid-generation some new programming had to be done since appropriate tools are not yet on the market. Of course a certain number of routines approved in batch operation could be integrated.

3.1 Geometry Preparation

The expenditure of necessary geometry editing depends on the form of the basic geometry, to be supplied is in any case one dataset per block, which contains all necessary geometrical data for the grid-generation. For handing over the data, an interface was defined, where all the information about the geometry is traced back to the most simple element - the 3D-point. All block edges as well as possibly necessary surface lines will be transfered in form of identified point sequences. All actual examples were developed within CADAM by means of 3D-splines - however handing them over was done also exclusively with point sequences. With this reduction no additional specifications about the geometry type is necessary. The system is open thereby for coupling to any CAD-system, and in addition, to the transfer of geometries, developed originally with help of closed functions, model-picked or NC-data, output of digitizers or data coming from a drawing board. Handing over geometries is independent of curve- or surface-algorithms. Only a few input conventions are necessary to identify the point-sequences.

3.2 Description of the Topology

A prerequisite for the correct interpretation of the transferred data is a unique relation between the counter directions I, J, K and the 6 blocksides. Consequently, for each point sequence the following must be specified:

- the block affiliation (block number),
- a characterisation, for block edges it is a side affiliation, surface lines additionally need specification of counting direction and position within a set of lines,
- finally the number of points.

A consistent order of the points within a sequence is assumed. The conventions at the interface between geometry preparation and grid generation is illustrated in Figure 12.

3.3 Mesh Generation

During the process of mesh generation first of all the point sequences are used as input to evaluate 3D-parametric cubic splines. After establishment of the appropriate coefficients, calculation of a first surface grid is done using an algorithm for the redistribution of points on given curves. This is done on all block surfaces twice in different counting directions. The used routine was developed originally for the generation of NC-data but it proved to be very flexible in its application.

If certain peculiarities of this approach are considered already when subdividing the total configuration into blocks, this starting solution will be already good enough in most cases so that it can be used directly as a final mesh.

If the obtained results of the redistribution are not satisfying, a Poisson-solver with variable source strength is available. Hereby the grids of the block surfaces can be adapted corresponding to the given block edge lines.

After completion of the (block-) surface grid, discretization of the volumes is done by means of the same procedures. Within integer-planes (index I, J or K = const.) starting solutions are established, which are optimized alternatively by redistribution or use of the Poisson-solver.

The whole process runs interactively and menue-driven at a graphic screen, all steps between basic geometry and final volume distribution can be repeated or varied or cancelled. The results of each action can be controlled and improved immediately if necessary.

Advantages of the used approach are:

- minimal computational expense, whereby an interactive operation is enabled,
- a very good reproduction of the described surfaces also of complicated configurations using the redistribution procedure,
- high flexibility within block-arrangement, i.e. arbitrary structuring for complex configurations is supported,
- easy handling of various grid specialities, for example the bisection of the meshwidth when passing to an adjacent block with use of multigrid logics within the solver.

3.4 Visualization

The visualization of the established grids is mentioned separately, because here a further, substantial advantage of the interactive approach becomes obvious. Even the illustration of block surface grids only in batch operating already requires several individual plotjobs - rotations in space and selection of the surfaces to be shown usually is done in a time consuming trial-and-error procedure. Sequences of plotjobs finally become necessary, if one tries to represent also the volume grids.

Use of a workstation with local intelligence (e.g. special processors for translation, rotation, scaling and clipping) will enable working at a continuously rotating wire frame model, where arbitrary parts can be shown or no-shown, so that at each time a complete overview of the current status is guaranteeed. The whole volume grid can be downloaded onto the workstation and represented there e.g. by a set of integer-planes. If their visibility is coupled with a suitable criterion to a valuator, then the user can walk through the volume-grid step by step and thereby gain a good impression of the cell distribution.

The process of visualization and control of the grids, which is necessary in any case, is reduced hereby to a fraction. Without these supplementary aids it is tedious and often only possible in iterative manner.

3.5 Examples

The first example in Figure 13 - 15 shows a configuration where the grid has been built up by a global H-H-structure of 3x3x4 blocks. Three of these - around, in front of and behind the store are replaced by a local H-O-mesh with 5x5 blocks. The combination of two different grid types in that case provides a good geometry representation of the wing as well as of the pylon and the external store. Results of Euler-computations using the shown grid are are given in [6]. The second configuration, given in Figures 16 - 18 consists of a combination of internal and external flow in the case of a fuselage with a sidemounted inlet, whereby the channel is also modelled up to the compressor entry plane. In this case a H-O-structure was used, 18 blocks with a total number of 236.000 volume cells are forming the computational grid. Computational results and a comparison with experimental data is discussed in [7].

3.6 Necessary Hard- and Software

The program development described here as well as the presented examples were carried out on a SPECTRAGRAPHICS 1500 workstation. Both main parts of the grid-developement can be accomplished at the same screen using different possible operating modes of the equipment. In the so-called emulation mode (unit operates like an IBM 5080) the basic geometry is established by means of the commercial software package CADAM. The mesh generation takes place in the native mode by means of special application programs, which permit the direct access to the graphic abilities of the equipment with the device-specific soft- and firmware called PRISM.

As far as within the first step commercial CAD-softwarepackages are used, because of the reduction of the interface data to point sequences, any similar systems could be coupled to the method.

Concerning the necessary application software within the second section, there are several other 3D-extensions of GKS (the GRAPHICS KERNEL SYSTEM) available, but each package is restricted to its special corresponding hardware.

Thats why some standards would be desirable, which enable an easy transfer of a non-trivial graphic-interactive application program from one workstation to another.

4. Solution Adaptive Meshes

In numerical fluid dynamics the equations governing fluid motion are often approximated by the means of difference equations, solved at discrete locations in the finite problem space. Associated with these approximations is a certain amount of numerical error (e. q. truncation error) which we desire to keep as small as possible. In general, if the higher order derivatives associated with truncation errors are negligible, then the error itself is negligible. If this is not the case, then the step size between adjacent points must be decreased.

Numerical solutions of the Reynolds averaged Navier Stokes equations require a very fine grid resolution in all those regions where viscous effects are dominating, as long as no wall functions are used. For flow fields with large separated regions which very often are highly influenced by those separated regions, it is impossible to prescribe correct wall functions. It is also impossible to predict the position and the shape of all the separated regions and the position of all the free shear layers. So at the grid generation for such flow fields the regions, where very fine grid resolutions are needed, are still unknown. If constant step sizes are used, this means an increase in the number of grid points over the entire space, which for most problems becomes prohibitively expensive. Some other, more practible solutions to this problem are:

- The use of local grid refinement. This approach uses a coarse global grid with embedded fine sub-grids in regions of interest, which is principally possible within the framework of the block structured concept.
- The use of solution adaptive grids. In this approach, the grids are adapted to the solution during the solution process.

In the following sections there are described two different methods for the generation of adaptive single block meshes and adaptive block structured grids with local grid refinement.

4.1 Mesh Adaption in Single Block Meshes

If the computational grid is adapted to preliminary results in such a way as to minimize the aforementioned error term, we can expect the final solution to be an improvement in terms of accuracy over the solutions obtained in uniform or arbitrary grids. In addition, one would expect the same accuracy for this, a so called "solution adaptive grid" as for a uniform grid having many more ponts.

It is assumed that the redistribution of grid points should be based on the distribution of the curvature of a typical, the flow field describing function u (for example: surface pressure distribution). The curvature is obtained at each point i by the central difference approximation

$$a_i = u_i = \frac{2}{h_3}\left\{\frac{u_{i+1} - u_i}{h_2} - \frac{u_i - u_{i-1}}{h_1}\right\} + O\left(h^2, h_2 - h_1\right) \tag{2}$$

using forward and backward difference operators. For sake of simplicity we may set $a_1 = a_2$ and $a_N = a_{N-1}$. By nomalizing the curvature with the constant step size h,

$$h = \frac{x_N - x_1}{N - 1} \tag{3}$$

we obtain a weighted measure k_i of curvature at each point:

$$k_i = a_i \frac{h_i}{h} \tag{4}$$

with

$$h_i = x_i - x_{i-1}. \tag{5}$$

In order to damp extreme values in curvature and to increase the interval of influence, a new measure of curvature,

$$\alpha_i = \frac{1}{2n+1}\sum_{j=2}^{n} k_{i+j-n}, \quad i = n+1,\ldots\ldots,N-n, \tag{6}$$

is introduced for inner points. At boundaries a similar but one-sided formula is used. In all cases described here, a value of n = 1 was used, resulting in smoothing three points.

The transformation function is finally obtained from the integration of alpha (see Figure 19):

$$S_i = \sum_{j=2}^{i} \alpha_j, \tag{7}$$

with $S_1 = 0$. One notices that the transformation function $S(x_i)$ has its maximum slope where the curvature of $u(x_i)$ has its maximum curvature, and its minimum slope where the curvature of u(x sub i) is also minimal. The table of values obtained from $S_i = S(x_i)$ can also be used in its inverse form $x_i = x(S_i)$. By dividing the interval

$$S_N = \frac{1}{h}\int_{x_1}^{x_N} \alpha_i dx \tag{8}$$

into N -1 subintervals,

$$S_i^+ = s_n \frac{i-1}{N-1}, \quad i = 2,3,\ldots N,$$

one can obtain through interpolation the new distribution $\bar{x}_i = \bar{x}(S_i^+)$. In order to guarantee monotonicity this interpolation must be linear, then from the existence theorem the inverse function exists because S_N is continous.

The new step sizes found by the procedure just described depend completely on the behaviour of the function $u(x_i)$. If this function is piecewise linear, some of the α_i become zero. This can lead to uncontrollably large step sizes. Since however, the accuracy of numerical methods always depends on the chosen step size, an additional condition must be introduced, controlling the maximum interval between two adjacent points. The step parameter P is defined as

$$h_{\max} = P\,h \tag{9}$$

Where h is again the step size for uniform point distribution. The gradients of S(x) are now compared against a minimum value

$$q = \frac{S_N}{(n-1)h_{\max}} \neq 0 \tag{10}$$

which is controlled by P. Therefore it proves neccessary to use an additional linear transformation in order to ensure such a minimum gradient of value q.

Figure 20 shows an example for this adaption techique for a C-type mesh around an airfoil.

The initial point spacing (lower mesh in Figure 20) is already non-uniform, having more concentrated points at the leading and trailing edges; in these regions a pressure distribution is assumed *a priori* showing larger curvature. The adapted grid in the upper part of the same Figure is based on the surface pressure distribution calculated by means of the initial mesh. Therefore concentrations of mesh points at the approximate middle of the upper and lower surface as well as at the trailing edge are due to the curvature of the pressure distribution. The influence of the grid adaption on the flow solution is given in reference [8].

4.2 2-D Adaptive Block Structured Grids with Local Grid Refinement

The use of adaptive grids in combination with local grid refinement combines the advantages and cancels the disadvantages of each method. So the use of adaptive grids requires a high number of grid points to avoid jumps in the grid spacing. On the other hand, the use of fine subgrids would be a very good approach for viscous flows, if the boundaries of those subgrids could be adapted to the structure of the flow field. If additionally a block structure, which is adapted not only at the geometric requirements but also at the structure of the flow field, is used, we will have a very effective discretization of the flow field; (adaptive grids with local grid refinement) and also a very effective procedure for the flow solution by the use of zonal approach (Euler/Navier Stokes) which due to the block structure can be done very simply.

The basic idea for the method described here has been given by J. Thompson in [9]. Following a method for generating 2-D adaptive grids with local grid refinement for Navier Stokes calculations is described.

A constant dicretization along the i direction in the index space is discribed by the relation:

$$\Delta x_i = const.$$

Which can be written in the computational (index-) space as:

$$x_\xi = const.$$

or

$$x_{\xi\xi} = 0$$

which is a one dimensional Laplace equation. It can easily be seen, that the relation for a two dimensional, constant discretization is:

$$X_{\xi\xi} + X_{\eta\eta} = 0$$

where X = (x,y) are the cartesian coordinates. If not the geometric distance but the product of a weighting function and the geometric distance is kept constant in the discretization, this can be expressed by the relation:

$$W\Delta x_i = const.$$

Where W is any weighting function. In the computational space this yields to :

$$W x_\xi = const.$$

or

$$W x_{\xi\xi} + W_\xi x_\xi = 0 \tag{11}$$

which is a one dimensional Poisson equation. Again, the relation for 2 dimensions is:

$$W_i X_{\xi\xi} + W_j X_{\eta\eta} + W_{i\xi} X_\xi + W_j X_\eta = 0 \tag{12}$$

Where W_i and W_j are the weighting functions for the two computational coordinate directions. The above equation is an elliptical partial differential equation which is commonly used for grid generation. If now characteristic properties of the flow field are taken as weighting functions W_i and W_j, the above PDE will generate adaptive grids. The weighting function for the computational i-direction should be coupled with the pressure distribution, and the weighting function in j direction, which is the direction normal to the main flow direction, should be coupled with any indicator for viscous effects. Numerous experiments with different weighting functions have shown that the best weighting function for the computational i-direction is given by the relation:

$$W_i = \alpha \frac{\partial p}{\partial x} + \beta \frac{\partial^2 p}{\partial x^2} \tag{13}$$

So the weighting function is a combination of the first and the second derivative of the pressure distribution. This gives a grid adaption to pressure gradients and extreme values. α and β are weighting parameters by which the user can make the first or second derivative more dominating. Both derivatives are normalized in such a way that the absolute values move between 0 and 1.0. In the j - direction, possible weighting functions may be the total pressure loss- or the vorticity distribution. It was however found out, that the total pressure loss is the most suitable parameter to drive the grid adaption to any flow field discontinuity, because its values move within a small range whereas the values of the vorticity spread over several powers of ten. So the weighting function for the j-direction has been choosen as:

$$W_j = \gamma (1 - \frac{p_t}{p_{t0}}) \tag{14}$$

Where γ again is a scaling parameter. The grid adaption can be performed in 3 levels:

- Adaption of the surface point distribution along the surface to the surface pressure distribution.
- Adaption of the field grid points normal to the flow direction.
- Adaption of the field grid points in flow direction.

The perimeter adaption along the surface is done by the solution of a one dimensional poisson eqation

$$\frac{W}{W_\xi} x_{\xi\xi} + x_\xi = 0 \tag{15}$$

If the above equation is approximated by finite differences in the index space this leads to a simple tridiagonal equation system. The weighting function is given by equation (13). For this surface pressure adaption, only the surface pressure distribution is required. For the field adaptions the weighting functions according to eqs. (13) and (14) are taken. Those weighting functions are introduced as source terms into the elliptical PDE for grid optimization. In order to get smooth adapted grids across the block boundaries, the field is divided into sub-regions which are as large as possible and which are overlapping. Only at the end of the grid generation process, the grid is split up into the final block structure.

The local grid refinement is treated as follows: First the uniform, finest grid is generated. Then the coarser blocks are obtained by dropping each 2., 4., 8. gridpoint in i- and/or j-direction. Figure 21 shows such an unadapted grid. Here again, the initial grid is already non-uniform. Along the surface the grid points are concentrated at the leading edge and at the trailing edge. For the grid adaption, the weighting functions of the flow solution are interpolated into the uniform fine grid. Then the grid adaption is performed for the uniform fine grid and finally the coarse subgrids are generated. The best strategy for the use of such adapted block structred grids seems to be the following:

- Make an Euler calculation in a coarse mesh to get the significant surface pressure distribution. (Position of the extreme values and gradients). It is not necessary that the solution is converged, it is only important, to have a significant pressure distribution.
- Next, a Navier Stokes grid adapted to this surface pressure distribution is generated. This grid can have coarse mesh sizes in j-direction in order to accelerate the time development of the solution.
- Start the Navier Stokes solution.
- During the solution process, the field grid is adapted from time to time by the use of the total pressure loss as weighting function. So the grid points are automatically concentrated in regions with highly dominating viscous effects.

Figures 22 and 23 show a significant pressure distribution and the total pressure loss contours which are obtained during the solution process for the geometry of Figure 21. In Figure 24 the surface pressure adapted grid is presented. Compared with Figure 21, it can be seen, that the grid points are concentrated in regions with gradients and with extreme values. Figure 25 finally shows the adapted grid. Now the viscous regions can be recognized in the grid.

The field adaption to the total pressure loss distribution is very stable and can be done automatically during the flow solution. It was also found, that the field adaption to the field pressure distribution has no advantages as long as there are no pressure discontinuities in the flow field. The adaption of the grid to the surface pressure distribution is sufficient and can be done once at the beginning of the calculation.

5. Conclusions

Although all the presented grid generation techniques use only elliptical grid generation (hyperbolic and parabolic grid generation is also widely in use), they show already, that there is no unique grid generation technique.

All automatic grid generation procedures have the advantages that the grid can be described by a few grid generation parameters and by this, the complete grid can be modified or changed very fast. But the automatic grid generation has its limitations in the complexity of the geometry. For each new geometry, the automatic grid generation procedure has to be extended to the new configuration.

The graphic interactive grid generation avoids the difficulties with the complexity of the geometry. On priniple, each grid can be "constructed" by the user, where automatic subsystems (algebraic or elliptical grid generation techniques) can be used.

Two- and three-dimensional viscous flow computations of complex configurations require a very large number of mesh points to resolve all the gradients properly. Here the block structuring in combination with mesh concentration and adaptive grid generation can help to provide the required flow field accuracy.

6. *References*

1. *J. F. Thompson, F. C. Thames, C. W. Mastin*
Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing any Number of Arbitrary Two-Dimensional Bodies. Journal of Computational Physics, Vol 15 1974

2. *S. Le.cher, W. Fritz, J. Grashof, J. Longo*
Mesh Generation Strategies for CFD on Complex Configurations. Paper in Lecture Notes in Physics Vol.170 Springer Verlag 1982

3. *W. Fritz*
Numerical Grid Generation around Complete Aircraft Configurations.
AGARD-CP-412 Paper 3 1986

4. *P. E. Rubbert and K. D. Lee*
Patched Coordinate Systems.
1982, Numerical Grid Generation edited by J. F. Thompson, North-Holland Puplishing Co., New York, pp. 235-252.

5. *S. Leicher, W. Fritz*
Numerical Simulation of 3-D inviscid Flow Fields around Complete Aircraft Configurationd.
ICAS Paper ICAS-86-1.3.2 1986

6. *S. Leicher, W. Seibert, B. Wagner*
Aerodynamische Waffenintegration
Dornier Report BF 36/85 B March 1985

7. *W. Seibert, S. Leicher, J.Grashof, B. Wagner*
Pitot Seiteneinlaun EM-2
Dornier Report BF 34/85 B December 1985

8. *W. Haase, K. Misegades, M. Naar*
Adaptive Grids in Numerical Fluid Dynamics
International Journal for Nummerical Methods in Fluids, Vol. 5, pp 515-528 (1985)

9. *J. Thompson*
Grid Generation
Lecture at Navier Stokes Flow Simulations, AIAA/VKI Study Seminar September 15-16, 1986

7. Figures

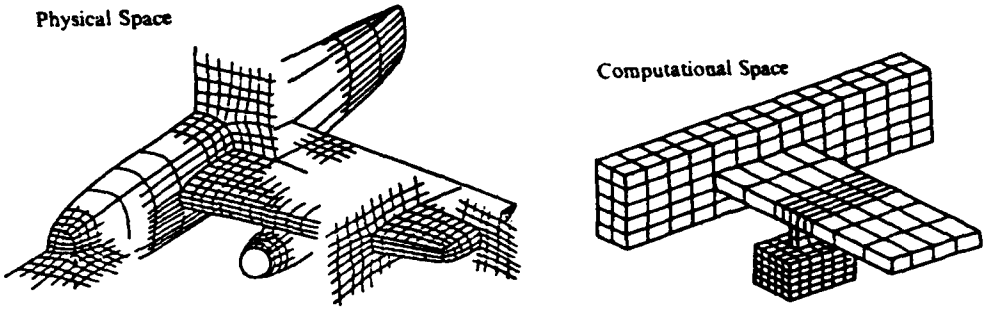


Figure 1: Block Structuring of a Complex Configuration

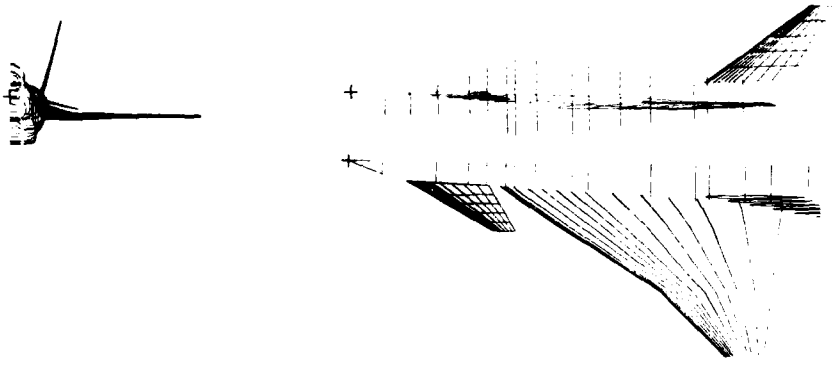


Figure 2: Geometry Definition of a Fighter Type Aircraft

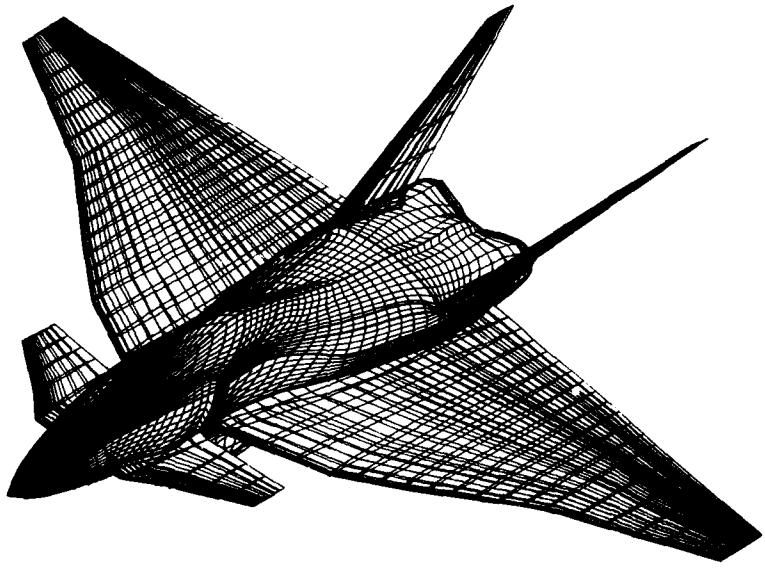


Figure 3: Surface Grid for a Fighter Type Aircraft

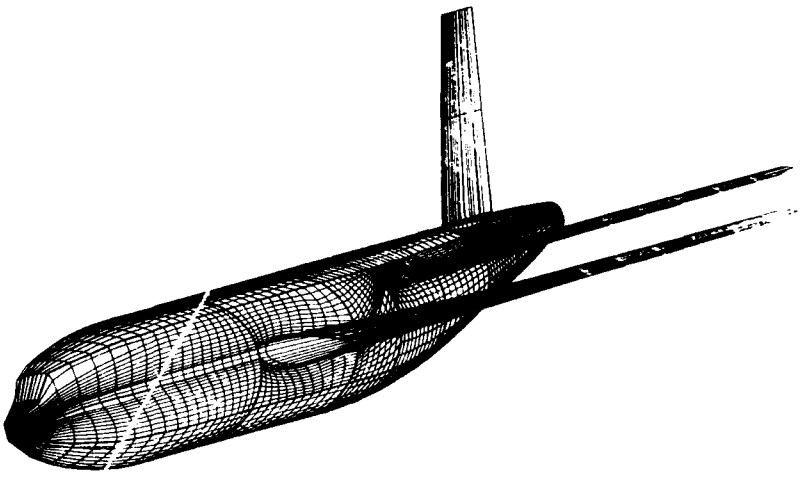


Figure 4: Surface Grid for a Transport Type Aircraft

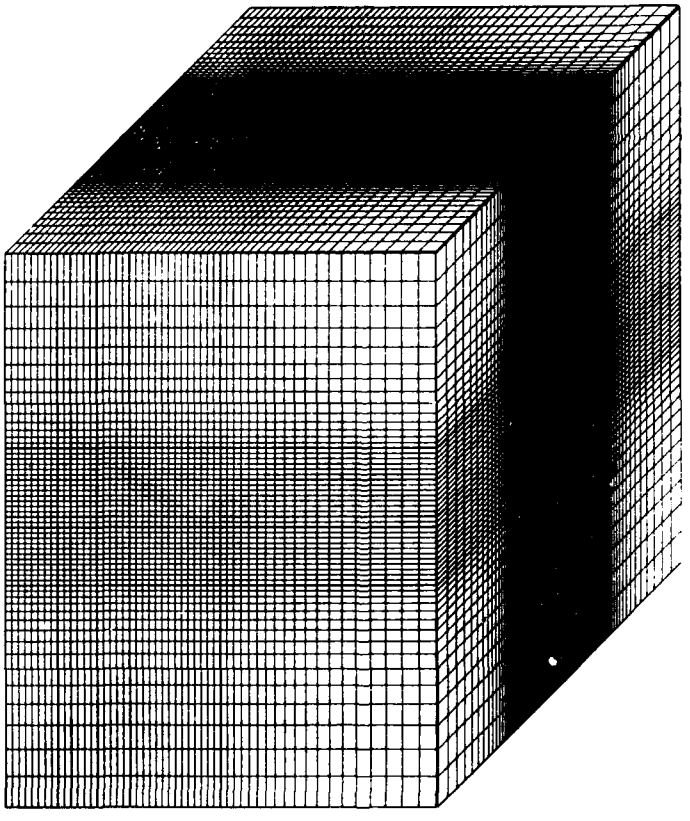


Figure 5: Sectionwise Storage Arrangement of the Complete Grid

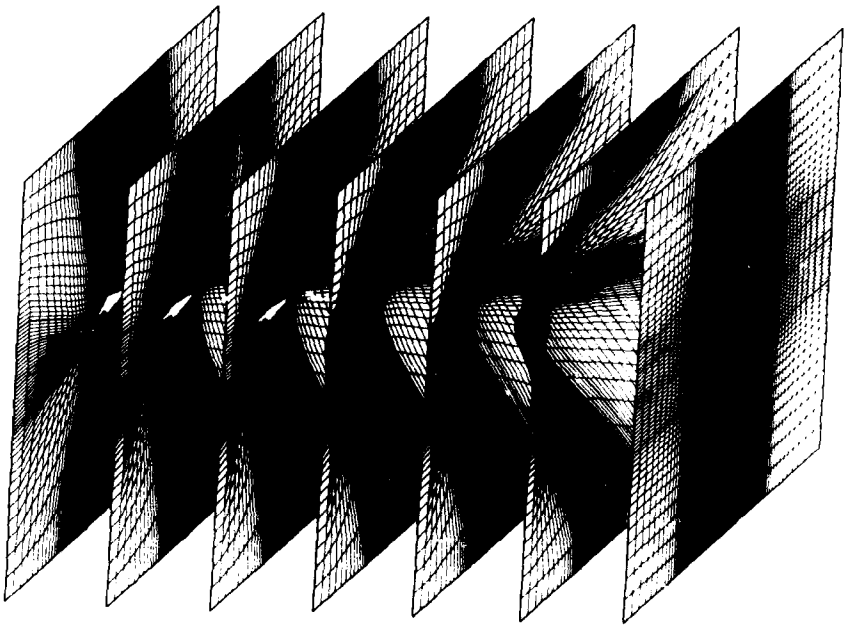


Figure 6: *Block Surfaces k = const.*

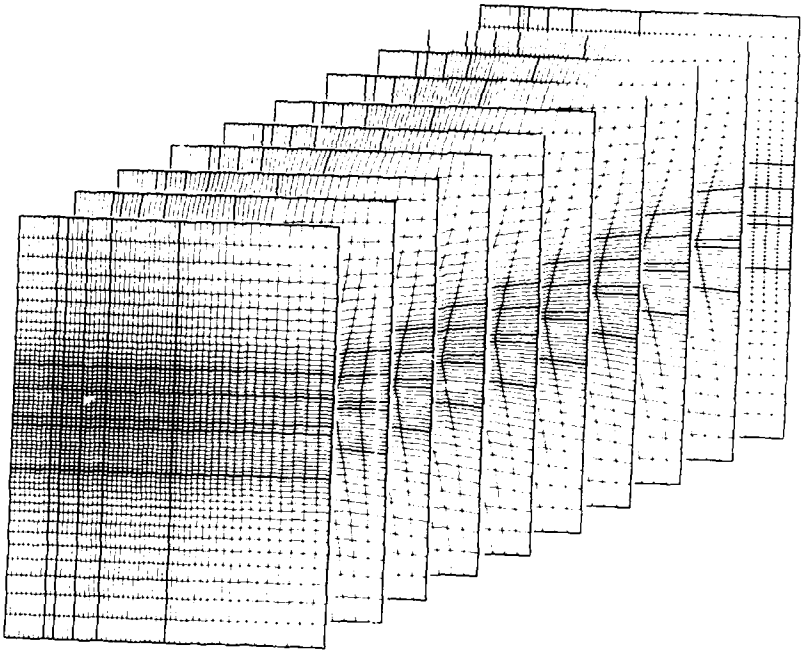


Figure 7: *Block Surfaces i = const.*

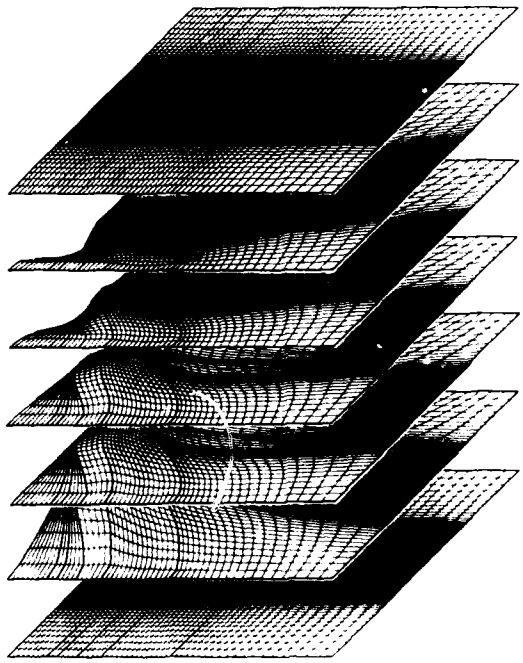


Figure 8: Block Surfaces j = const.

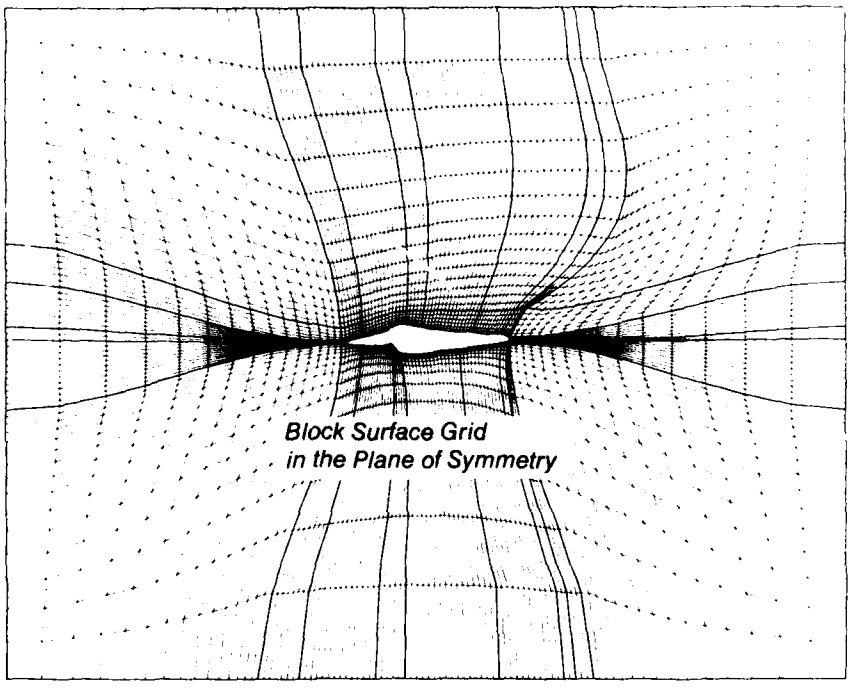


Figure 9:

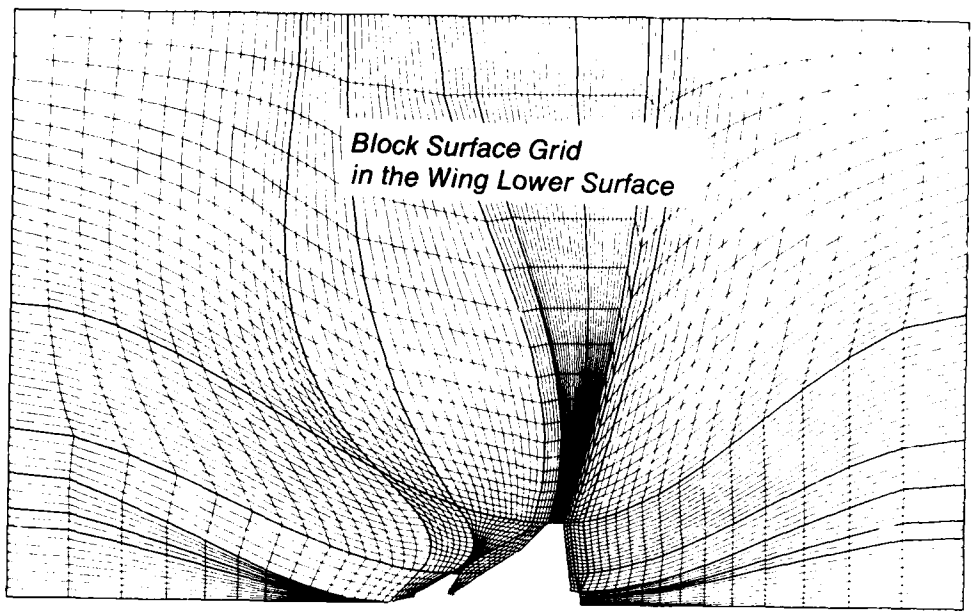


Figure 10:

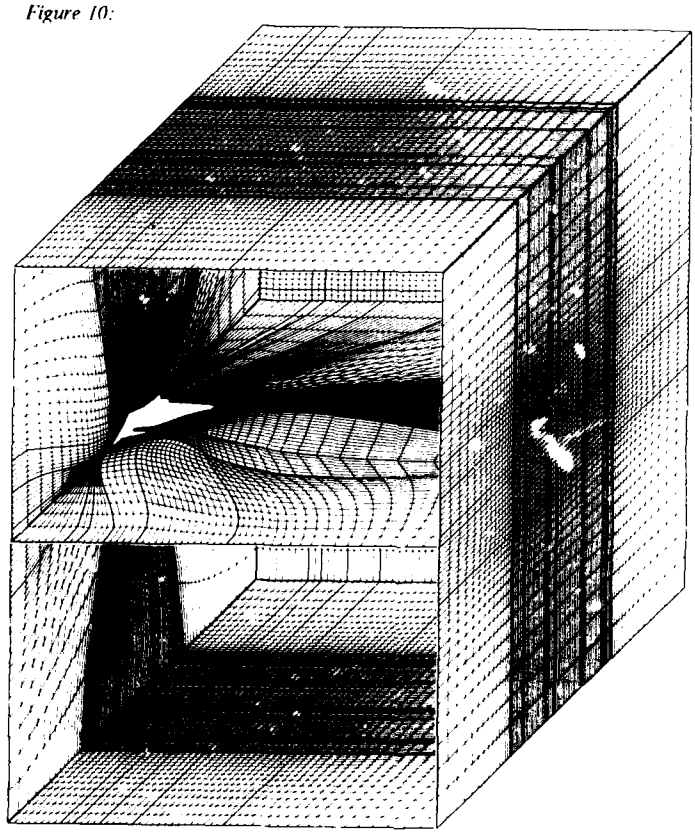


Figure 11: 3-D Grid Arrangement

A Counter directions I, J, K and identification of the block-sides

B Characterisation of the block-sides

C Characterisation of the surface-lines

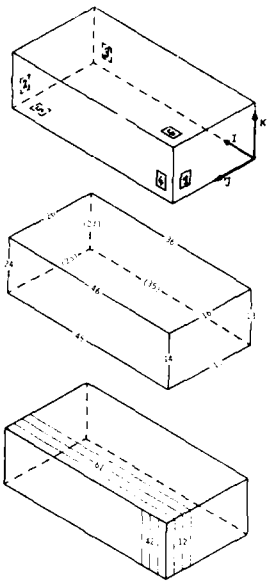


Figure 12: Conventions at the Interface between Geometry Preparation and Grid Generation

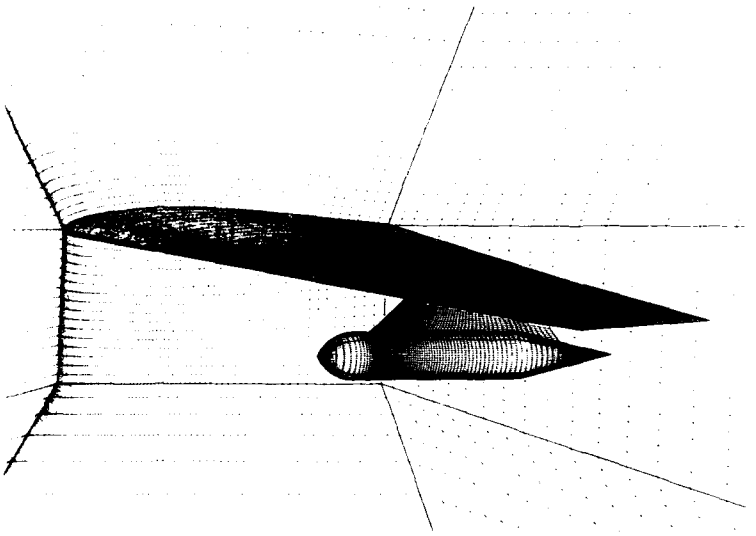


Figure 13: Wing-Pylon-Store Combination, Surface Mesh and Plane of Symmetry

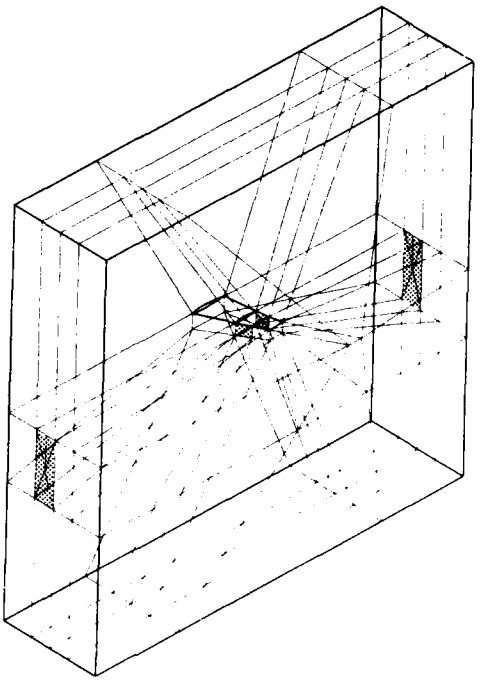


Figure 14: Wing-Pylon-Store Combination, Block Structure of Global Arrangement

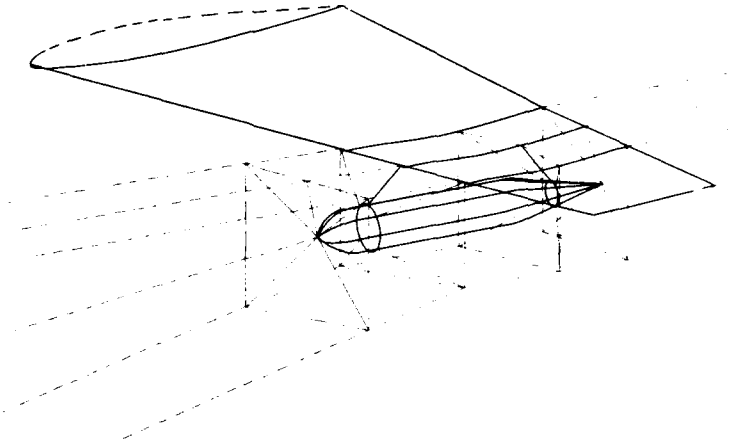


Figure 15: Wing-Pylon-Store Combination, Local Block Structure

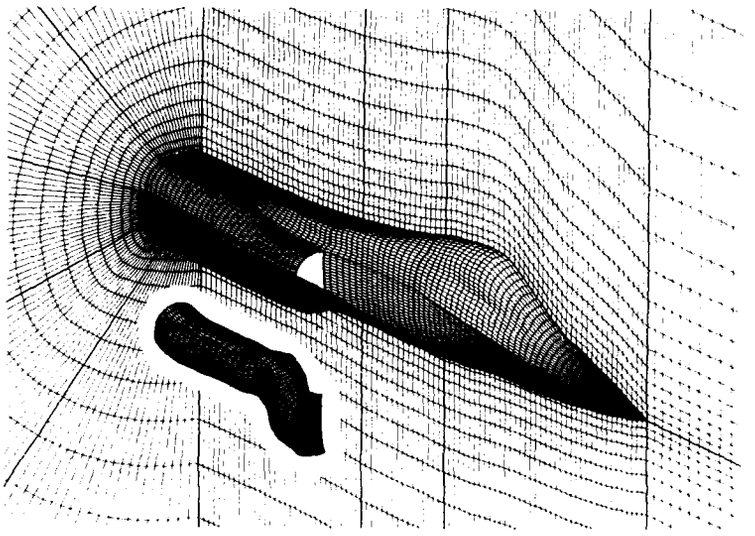


Figure 16: Fuselage with Inlet, Mesh on Body Surface, Plane of Symmetry and within Engine Channel

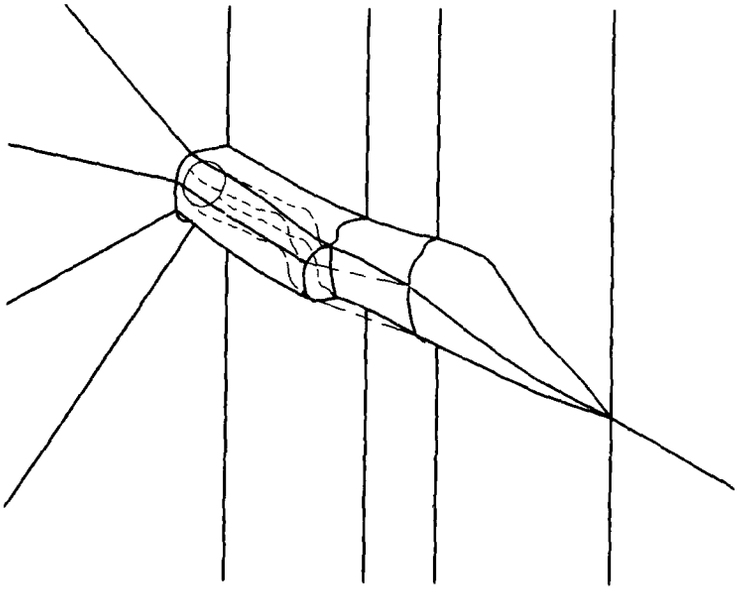


Figure 17: Fuselage with Inlet, Block Boundaries near Inlet

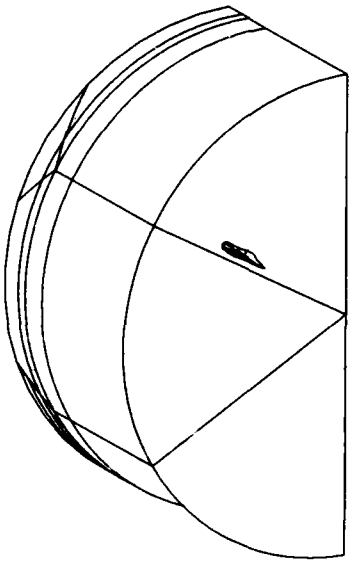


Figure 18: Fuselage with Inlet, Block Boundaries at Far Field

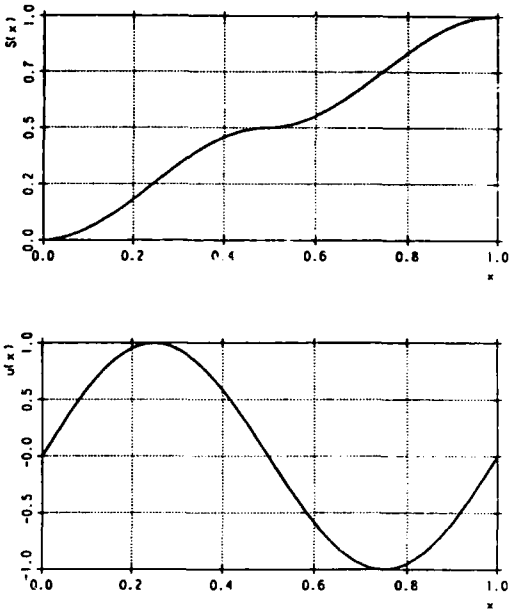


Figure 19: Distribution of aFunctiort u(x) and the Appropriate Transformation Function S(x) Normalized to Unity

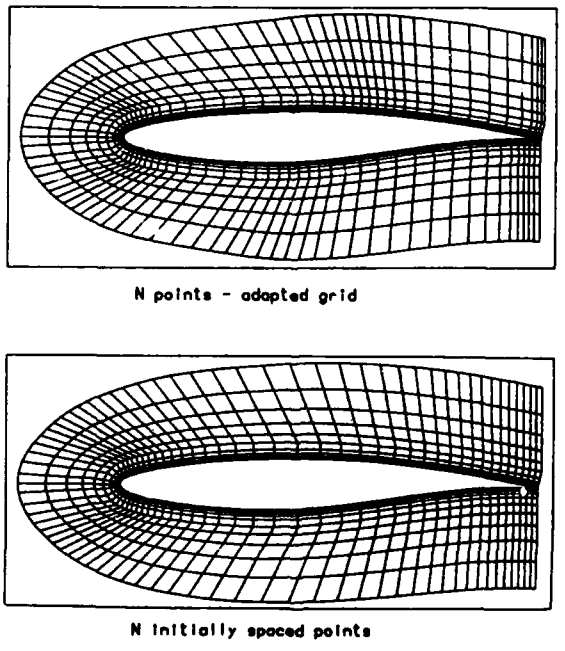


Figure 20: Grid Structures for RAE 2822 Using 83 Points for Surface discretization in a Single Block Mesh

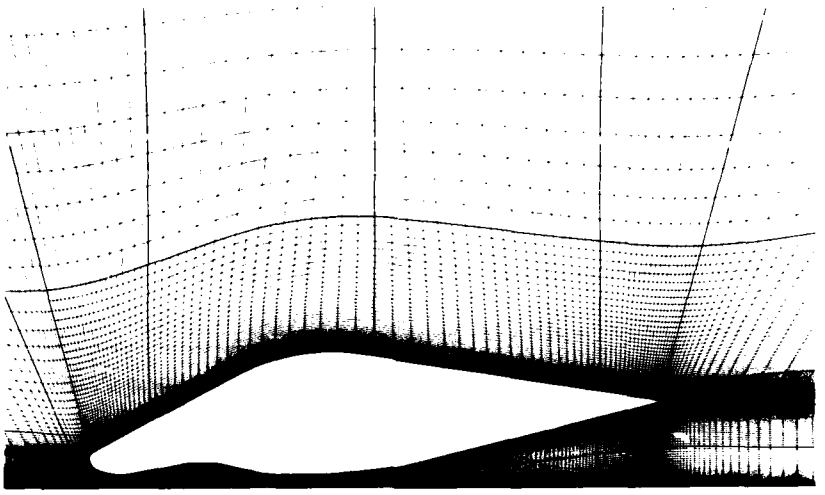


Figure 21: Non-Adapted Block Structured Grid

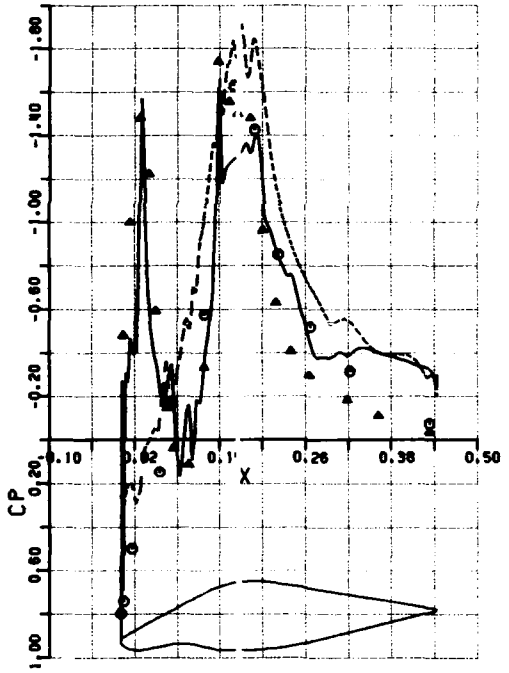


Figure 22: Significant Surface Pressure Distribution.

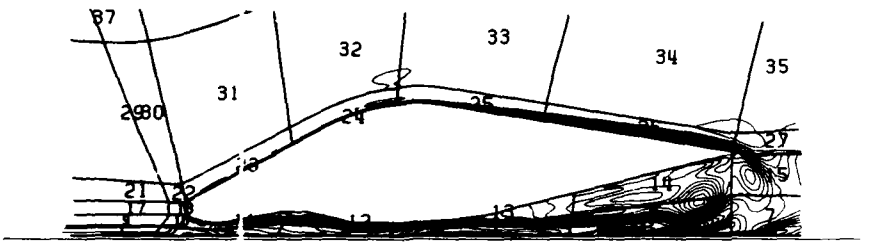


Figure 23: Total Pressure Loss Contours.

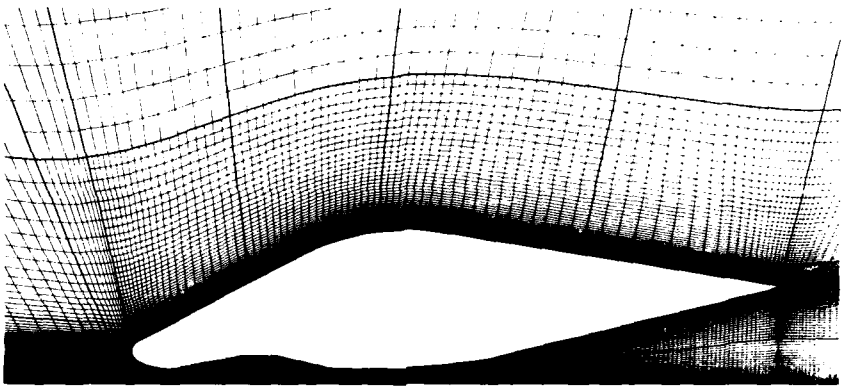


Figure 24: Surface Pressure Distribution Adapted Grid.

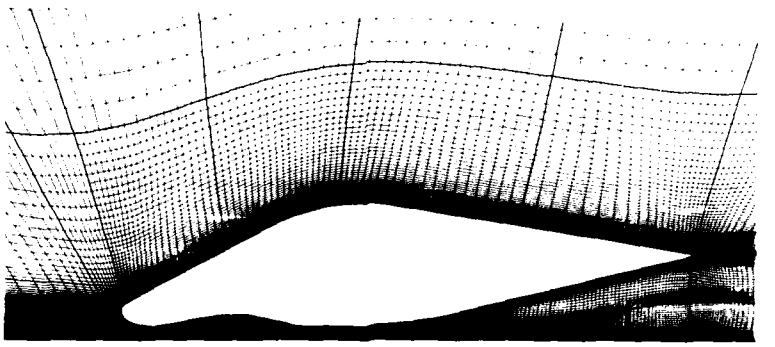


Figure 25: Surface Pressure Distribution and Total Pressure Loss Adapted Grid.

4.12 EXPERIENCE WITH THREE-DIMENSIONAL COMPOSITE GRIDS*

by
J. A. Benek, T. L. Donegan, and N. E. Suhs
Calspan Corporation/AEDC Division
Arnold Air Force Station, Tennessee 37389-9998

ABSTRACT

Experience at the AEDC with the three-dimensional (3-D), chimera grid embedding scheme is described. Application of the inviscid version to estimate wind tunnel wall interference on a wing/body/tail configuration is described. Applications of the viscous version compute a 3-D cavity and a multiple-body configuration. A variety of grid generators is used, and several embedding strategies are considered.

1.0 INTRODUCTION

In the last ten years, Computational Fluid Dynamics (CFD) has evolved from an academic enterprise into a necessary, if not integral, part of aircraft design and development. Two circumstances have stimulated this change: the maturation of fast numerical algorithms for solution of the Euler and Navier-Stokes equations and the reduction of the price of the large supercomputers required to perform the computations. As the entry costs decrease and the value of flow simulations becomes more widely recognized, the demands for even more complex simulations increase. The heightened level of expectation also increases pressure to produce "timely" solutions. This pressure can only be expected to increase as CFD becomes more closely coupled to the design and development processes. Frequently, the most critical phase in meeting the demand for computations is the construction of a suitable mesh. To ameliorate the difficulties experienced with grid generation, alternative computational strategies are being explored. Basically, they can be divided into two categories: global approaches and domain decomposition approaches.

The global mesh approach uses a single computational net to discretize the geometry and flow field (e.g., Thompson (Ref. 1), Rubbert and Lee (Ref. 2), and Shang and Scherr (Ref. 3)). Complex geometry frequently requires the introduction of internal boundaries (e.g., cuts) into the domain and may result in very skewed grids and regions of unacceptably low spatial resolution. The introduction of internal boundaries increases the bookkeeping required in the flow solver and can require modifications to the solution algorithm. One novel approach utilizing a global mesh is described by Jameson, Baker, and Weatherhill (Ref. 4). The major thrust of this work is to use a finite volume algorithm based on tetrahedrons and eliminate the requirement for an ordered mesh. A complex data-structure is required to define the relationships among the grid points comprising the volumes.

Domain decomposition includes many techniques: zonal or grid patching [e.g., Hessenius and Pulliam (Ref. 5), Rai (Ref. 6), and Holst, et al. (Ref. 7)], and grid embedding/oversettings [e.g., Atta and Vadyak (Ref. 8), Benek, et al. (Ref. 9), Venkatapathy and Lombard (Ref. 10), and Berger (Ref. 11)]. The basic idea of this strategy is the subdivision of the computational domain into regions (not necessarily disjoint) that can be more easily meshed. An additional advantage is that each subdomain may be treated separately and a different flow model or solution algorithm used in each. Such flexibility provides economies in computer resources as the more expensive viscous flow solvers can be confined to regions where viscosity dominates the flow. The key to successfully implementing this strategy is provision of a means of intergrid communication. This is the point at which the various techniques differ most widely. All these techniques require additional bookkeeping beyond that required for the basic flow solver to facilitate communication.

Presently, no one method has been demonstrated to be clearly superior. It seems likely that some synthesis of the various strategies will become the method of choice. In the meantime, we have chosen the grid embedding approach as it includes grid patching as a special case and thus provides a flexible method for accomplishing a broad range of flow simulations. In this paper we will describe our experience with the chimera scheme which was first developed by Benek, Steger and Dougherty (Ref. 9). The three-dimensional, color graphics code required to support this effort was developed by Buning and Steger (Ref. 12).

2.0 DESCRIPTION

The chimera grid embedding technique is a domain decomposition strategy and has two principal elements: (1) decomposition of the domain into subdomains which typically overlap and (2) communication among the grids. The selection of subdomains is arbitrary;

---

*The research reported herein was performed by the Arnold Engineering Development Center (AEDC), Air Force Systems Command. Work and analysis for this research were done by personnel of Calspan Corporation/AEDC Division, operating contractor for the AEDC aerospace flight dynamics test facilities. Further reproduction is authorized to satisfy needs of the U. S. Government.

the major considerations are the identification of regions that may be easily meshed, the isolation of special regions of the flow (e.g., where viscous effects are important), and the available computer memory (which determines the maximum number of points in each subdomain). Theoretically, this means the total number of mesh points in the entire domain is unlimited. Intergrid communication is established by the transfer of boundary data among the subdomain grids. The data for embedded grid boundaries are obtained by interpolation of the independent variables in the mesh in which the boundary is embedded.

There are two types of interpolation boundaries: (1) outer boundaries and (2) artificial boundaries. Artificial boundaries are produced whenever a solid surface is embedded in or overlaps another subdomain. Figure 1 depicts a flapped airfoil where the flap mesh lies within the airfoil mesh. Points of the airfoil mesh are contained within the solid boundary created by the flap surface, and therefore lie outside the computational domain. A portion of the airfoil mesh in the neighborhood of the flap is excluded from the airfoil grid (i.e., the shaded area around the flap within the airfoil mesh). The boundary of this excluded region of the airfoil mesh is an artificial boundary.

The computational procedure can be illustrated as follows: The solution is advanced on the airfoil mesh. Outer boundary data for the flap mesh are interpolated from the solution on the airfoil mesh and transferred to the flap solution. The transferred data are used as boundary conditions to advance the solution on the flap mesh. Data for the artificial boundary of the airfoil mesh (dashed line on the flap grid) are interpolated from the solution on the flap grid. The interpolated data are transferred to the artificial boundary in the airfoil mesh and the process repeats until convergence is obtained on each mesh.

The chimera procedure naturally separates into two parts, (1) generation of the composite mesh and associated interpolation data and (2) solution of the flow model or models on each mesh. Each part is embodied in a separate computer code, PEGSUS and XMER3D. PEGSUS takes independently generated component or subdomain grids and the embedding specifications as input and automatically constructs the composite mesh and computes the interpolation data which are output. XMER3D takes the PEGSUS output and flow specifications as input and solves the appropriate flow model on each grid.

2.1 PEGSUS

Automatic generation of a composite mesh from the input component grids requires PEGSUS to (1) establish the proper lines of communication among the grids through appropriate data structure, (2) construct holes within grids, (3) identify points within holes, (4) locate points from which boundary values can be interpolated, and (5) evaluate interpolation parameters. In addition, PEGSUS performs consistency checks on the interpolation data to assure their acceptability and constructs output files with the data structures appropriate to XMER3D. The most recent version of PEGSUS allows very general interactions among grids as indicated in Fig. 2. In addition, any grid may introduce a hole into any other mesh. Details of the hole construction process, and associated data structures, are provided by Benek, et al. (Refs. 9, 13, and 14). A trilinear interpolation is used to obtain boundary data.

2.2 XMER3D

The implementation of the chimera scheme must provide for the use of multiple flow models. The current choice of models is the 3-D Euler equations for inviscid flow and the 3-D thin-layer Navier-Stokes equations for viscous flow. The algebraic model of Baldwin and Lomax (Ref. 15) is used to simulate turbulent flow. The implicit, approximate factorization scheme of Beam and Warming (Refs. 16 and 17) is used to solve the model equations. The implementation follows that of Pulliam and Steger (Ref. 18) and uses explicit boundary conditions. Modifications to accommodate the chimera scheme are described by Benek, et al. (Ref. 14).

3.0 APPLICATIONS

A major motivation for the development of the chimera scheme at the AEDC was the requirement to provide routine computational support to testing. Estimates of the effects of the wind tunnel environment on aerodynamic data are of particular interest. Typically, lead times are short and grid generation is usually the pacing item in performing CFD simulations. Also, there is the requirement to compute time-dependent flows involving aerodynamic configurations in relative motion as exemplified by the space shuttle booster configuration and store separation from military aircraft.

The 3-D chimera scheme has been used to compute both viscous and inviscid flows over a variety of configurations. These include a wing/body/tail, bodies of revolution in close proximity, cavity flows, and base flows for Mach numbers spanning the range from subsonic to supersonic. The following sections will illustrate some of these applications of the chimera scheme.

3.1 Inviscid Flows

One of the intended uses of the chimera scheme at the AEDC is the computation of wind tunnel wall and support interference (e.g., Kraft, et al. (Ref. 19) and Suhs (Ref. 20)). A version of the chimera scheme was developed for this purpose. The model shown in Fig. 3 was designed for assessment of wind tunnel wall interference. It consists of

a blunted ogive-cylinder and a mid-mounted wing and tail. The wing and tail are constant chord planforms swept back at 30 deg and have no twist or taper. Cross sections parallel to the plane of symmetry are NACA 0012 airfoils. Initial, free-air solutions for the configuration were reported in Refs. 13 and 14.

For the tunnel calculations the outer boundaries of the grids about the fuselage, a portion of the sting support, wing, and tail for this model are illustrated in Fig. 4. The wind tunnel walls are represented as shown in Fig. 5 with the model embedded in the tunnel mesh. The region devoid of mesh lines on the tunnel symmetry plane in Fig. 5 represents the hole in the tunnel grid introduced by excluding points from the solution on the tunnel grid in the vicinity of the model.

Figure 5 illustrates the flexibility inherent in the chimera scheme. The model geometry and sting grids were constructed by adding a mesh containing the sting to an existing mesh used to model the fuselage. The component-by-component construction process is particularly useful for wall interference calculations because no additional grid generation is required to change model angle of attack. All that is required is that the grids representing the wind tunnel model be rotated relative to the tunnel mesh and be re-embedded in it. PEGSUS performs such transformations on component grids by a single change of input.

Several grid generators were used to construct the component grids shown in Figs. 4 and 5. They are a two-dimensional (2-D) grid generator developed by Sorenson (Ref. 21), and the three-dimensional generators developed by Soni (Ref. 22), and Thompson (Ref. 23). There are a total of 250,445 grid points in five meshes for this configuration.

The wall interference model was tested in the 1-ft Aerodynamic Wind Tunnel (1T) and in the 4-ft Aerodynamic Wind Tunnel (4T). Tunnel 1T has a one-foot-square test section, and 4T has a four-foot-square test section. The model has 2.5-percent blockage in 1T but only 0.16-percent blockage in 4T, so the 4T data are considered to be interference-free over the range of test conditions presented. Measured static pressure data were obtained on the model surface and at an interface near the Tunnel 1T walls at a radius of 5 inches (see Fig. 6). Static pressures along specific streamwise lines at the interface were measured by a two-component static pipe technique (Ref. 24). For the comparisons here, data were obtained at angular locations, θ, of 15, 85, 95, and 165 degrees as shown in Fig. 6.

Calculations were made to compare the computed pressures with the measured pressures and to determine the quantitative effects of wall interference on the model. The calculations were made using the chimera technique and the application at the tunnel walls of a porous wall boundary condition developed by Jacocks (Refs. 19 and 25). The conditions chosen for comparison were a Mach number, M_∞, of 0.9 and an angle of attack, α, of 4 deg. The tunnel porosity is uniform at three percent.

The pressure coefficient comparisons at the interface (Fig. 7) show good agreement and correctly predict the trends. The expansions near the wing and tail locations are evident, especially near the side wall. The wall interference effects on the model surface may be seen by comparing the computed and experimental pressure coefficient distributions on the wing, fuselage, and tail for both the tunnel and free-air cases (Fig. 8). The free-air solution computes a shock further downstream than the tunnel solution. This is consistent with the tunnel data, assuming the 4T data are interference-free. The absence of the viscous effects in the calculations result in the shock wave location being aft of the experimental shock. However, the trends are the same.

Mach number contours on the wall interference model are presented in Fig. 9. The contours join smoothly across mesh boundaries. The shock wave on the wing can be seen to continue around the fuselage. The figure illustrates the effect of decreasing spatial resolution in high gradient regions. The shock wave can be seen to be smeared on the fuselage compared to the wing because of the decreased resolution in the fuselage grid.

The success of the chimera scheme in providing realistic estimates of transonic wall interference has made its use for test planning and data analysis routine at the AEDC. Detailed descriptions of the wall interference calculations may be found in Ref. 26.

3.2 Viscous Flows

Cavity Flow

Interest in the flow in and around cavities has increased with the need for advanced aircraft to carry stores internally. Benefits from such configurations include increased range, better maneuverability, and reduced detection signatures (Ref. 27). Still, difficulties arise when attempting to safely eject a store from a weapons bay. In order to understand these difficulties a computational effort to determine the loads on, and trajectories of, stores in weapons bays is being pursued at the AEDC. As a first step, an empty 3-D rectangular cavity is modeled.

Using the grid overlap capabilities of the chimera scheme, two grids were developed to define the rectangular cavity in a flat surface. The first grid is a Cartesian grid defining the cavity and a region above the cavity. In Fig. 10, a sidewall plane of the cavity grid is shown. The cavity grid has a concentration of points along all solid wall surfaces and in the region of the shear layer. This grid extends above the cavity by 20 points in order to capture the entire shear layer in one grid. The cavity grid has a

total of 79,002 points. Also shown in Fig. 10 is a side plane of the Cartesian grid defining the region exterior to the cavity. Again, points are concentrated along the solid wall. This grid also extends in front of the flat plate in order to allow the flow to stagnate on the leading edge and allow a boundary layer to grow. The grid above the cavity has 78,625 points. These two grids overlap with a common region above the cavity and match point for point.

Comparisons of computations were made with experimental data taken at the Trisonic Gasdynamic Facility of the Air Force Wright Aeronautical Laboratories (Ref. 28). The Trisonic Gasdynamic Facility is a closed-circuit, continuous-flow wind tunnel with a two-foot-square test section. The Mach number range is 0.23 to 3.0. The rectangular cavity tested has a length-to-depth (L/D) ratio of 5.6 and a width-to-depth (W/D) ratio of 1.7. The dimensions of the cavity and flat plate are shown in Fig. 11. Pressure data were taken along the plane of symmetry of the cavity on the front, bottom, and aft walls, as well as on the sidewall and on the flat plate surface as illustrated by Fig. 11. Two types of data, steady and fluctuating static pressures, were made. The steady measurements were made using standard pressure transducers connected to static orifices and the fluctuating (or unsteady) measurements were made with flush mounted Kulite® pressure transducers. The unit Reynolds number was 2.31×10^6 per foot.

Experimentally, cavity flows have been shown to be unsteady with large temporal pressure fluctuations (Refs. 29 and 30). Therefore, the flow solver was run with global time stepping. The characteristic time for the flow, t_{ch}, is defined as the time for the flow to traverse the length of the cavity at free-stream velocity, approximately 0.85 ms. Typically, the calculation is run for 5 t_{ch} to permit the initial starting transient to decay. After the initial startup, the steady pressure coefficients are calculated from time-averaged pressures for the succeeding 6 t_{ch}. Comparisons of data and calculations for fluctuating pressures are made in terms of the sound pressure level (SPL). SPL in decibels (db) is defined as

$$\text{SPL (db)} = 180 + 20 \log (P_{rms}/P_{ref})$$

where P_{rms} is the root mean square of the pressure in psi and P_{ref} is 2.90075 psi, a standard reference pressure. As with the pressure coefficient results, the SPL is calculated over the same 6-t_{ch} interval.

In Fig. 12 calculated centerline pressure coefficients at $M_\infty = 0.74$ are compared to data for the front, bottom and aft walls of the cavity. The comparisons in Fig. 12 show good agreement between calculation and data. Of particular note is the good agreement on the aft wall where the shear layer stagnates. Comparisons of data and calculation for the SPL are shown in Fig. 13. The difference between calculation and data ranges from 2 to 5 db; still, the general trend is represented by the calculation.

In Figs. 14 and 15 representative details of the cavity flow are shown at four discrete time slices. The time differences between each time slice is 0.4 t_{ch}. In Fig. 14, Mach contours are shown for the cavity plane of symmetry. At $t = 11.0\ t_{ch}$, the shear layer across the cavity opening is stagnating on the back wall and is in the process of moving out of the cavity. The shear layer is shown to have moved out of the aft end of the cavity at $t = 11.4\ t_{ch}$. At $t = 11.8\ t_{ch}$, the shear layer begins to move back down into the cavity setting up a separation region downstream of the aft edge of the cavity. Finally, at $t = 12.2\ t_{ch}$, the shear layer at the aft end of the cavity has moved back into the cavity as the separation region past the aft edge increases in size. The shear layer at the front portion of the cavity shows relatively small changes in the flow.

Another way to look at this complex flow of the cavity is shown in Fig. 15. For this figure the mass flux across the cavity opening is plotted as a 3-D surface. If the surface bulges upward, mass is flowing out of the cavity. The three-dimensionality of cavity flow is illustrated by the changes in the mass flux distribution at different locations and times. The mass flux shown at the front edge of the cavity is caused by vortices that are generated in this region. With the exception of the front-edge complexity, the flow is shown to have a greater amplitude at the aft end of the cavity which is consistent with the large SPL distribution of Fig. 13. Details of this work can be found in Ref. 31 along with results for $M_\infty = 1.5$.

Three-Body Configuration

Flow about an aerodynamic configuration consisting of three identical bodies of revolution was computed. Each body (Fig. 16) consists of a 3.333-caliber cylindrical centerbody and a 1.667 tangent-ogive forebody and afterbody. The afterbody is truncated to join a 0.7-diameter sting. Details of the model and a discussion of the experiment are given by Cottrell, et al. (Ref. 32). The body axes are arranged in an equilateral triangle shown in Fig. 17. The spacing in the figure is given in model diameters. The right, left, and bottom designations are consistent with Ref. 32 and were established by looking upstream. The composite grid about this configuration consists of ten grids with a total of 627,172 points. The outer mesh G_1 is a hemispherical shell whose polar axis is the x-axis. Grids G_2, G_3, and G_4 (Fig. 18) are cylindrical grids whose axes coincide with the polar axis of G_1. These three grids have continuous grid lines and slopes across the grid boundaries. Each grid is "blocked" with two viscous grids. Grids G_5, G_7, and G_9 are hemispherical viscous grids representing the three forebodies. Grids G_6, G_8, and G_{10} are cylindrical grids representing the aft portion of the three bodies. Figure 18 shows the three bodies and their grids embedded within the three cylindrical grids, G_2, G_3, and G_4. Figure 19 shows a cross-section of the composite mesh in a vertical

plane through the x-axis (see Fig. 17). The overlap regions among the grids can be seen in the figure. The flow field was computed for the three-body configuration at $M_\infty = 0.95$, $Re_D = 2 \times 10^5$, and $\alpha = 0$ deg. This flow was assumed to be turbulent from the nose. The Baldwin-Lomax (Ref. 15) algebraic turbulence model was used to simulate the effects of turbulence.

Figure 20 shows axial distributions of C_p at several azimuthal locations on the lower body. Because of the 120-deg flow symmetry (Ref. 32) only one body need be examined. The agreement between the calculation and experiment is generally good on the forebody. The comparison becomes less favorable as separation is approached but again becomes generally good over the afterbody. The data indicate that the interior flow (i.e., within the "channel" formed by the bodies, e.g., $-60 \leq \phi \leq 60$ deg on the lower body) is accelerated compared to the exterior flow. The computation predicts the interior shock too far upstream and does a poor job of predicting the exterior separation/shock interaction. Such behavior is not unexpected as the Baldwin and Lomax turbulence model does not predict separated flows well, especially as the separation becomes massive. Modifications suggested by Degani and Schiff (Ref. 33) are being investigated to determine if they will improve agreement in the separated regions.

Figure 21 shows computed particle paths near the model surface and experimental oil pictures. Figures 21a and b compare "computed" and measured oil-flow patterns as seen from above, and Figs. 21c and d make a similar comparison as seen from the bottom of the configuration. In general, the basic features of the experimental oil flows are captured. Kaynak, et al. (Ref. 34) discuss the difficulty of comparing and interpreting oil-flow patterns and particle streamlines. We will not endeavor to analyze the flow patterns in more detail here.

The flowfield was also computed for the three-body configuration at $\alpha = 4$ deg and $M_\infty = 0.95$. These results are presented in Ref. 35 along with computational results for an isolated body at $M_\infty = 0.95$ and $\alpha = 0$ and 4 deg.

4.0 DISCUSSION

Sections 2 and 3 described our experience with the chimera scheme. However, there are several other aspects of its use that cannot be as clearly documented and several questions that remain unanswered. Perhaps, the most significant change that was made from the 2-D work reported by Benek, et al. (Ref. 9) was a change from the mixed 2nd/4th-order accurate approximations of Pulliam and Steger (Ref. 18) to a consistently 2nd-order approximation. Large oscillations in the solution with the mixed-order scheme occurred when grid boundaries crossed high gradient regions. Switching to a 2nd-order scheme has eliminated this problem.

Another question that commonly arises involves the interpolation at grid boundaries. Is the boundary approximation conservative? Our experience indicates that the major factor affecting accuracy at the boundaries is the resolution between the grids in the neighborhood of the boundary. Whenever there is a "large" mismatch in resolution, convergence slows and large oscillations in the solution are evident near the interface. Should the mismatch occur where the interface crosses a high gradient region, the situation is exacerbated. A more detailed and systematic study of this aspect of domain decomposition techniques is in order.

The chimera scheme was designed to function independently of the particular generation scheme used to construct subdomain grids. Our experience with grid generators include 2-D and 3-D elliptic codes, 3-D algebraic codes, and a hyperbolic code. The chimera scheme has successfully combined subdomain grids from several grid generators and many different topologies and has provided the basis for routine calculation of transonic interference effects from tunnel walls and model support structure at the AEDC.

SUMMARY

We have described our experience with the chimera grid embedding scheme. The method was applied to the computation of transonic wall interference with particular success and is being used routinely for support to testing at the AEDC. Experience with the viscous version is still being accumulated, but the potential to compute a wide range of flows has been demonstrated. Component grids have been generated by several 2-D and 3-D grid codes which employ algebraic and partial differential equations as generators. We experienced no difficulties combining grids constructed by the various methods into a composite mesh.

REFERENCES

1. Thompson, J. F., ed. Numerical Grid Generation. North-Holland, New York, NY, 1982.

2. Rubbert, P. E. and Lee, K. D. "Patched Coordinate Systems." Numerical Grid Generation, J. F. Thompson, ed., North-Holland, New York, NY, 1982.

3. Shang, J. S. and Scherr, S. J. "Navier-Stokes Solution of the Flow Field Around a Complete Aircraft." AIAA Paper 85-1509, July 1985.

4. Jameson, A., Baker, T. J., and Weatherhill, N. P. "Calculation of Inviscid Transonic Flow Over a Complete Aircraft." AIAA Paper No. 86-103, January 1986.

5. Hessenius, K. A. and Pulliam, T. H. "A Zonal Approach to Solutions of the Euler Equations." AIAA Paper 82-0969, June 1982.

6. Rai, M. M. "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations." AIAA Paper 84-0164, January 1984.

7. Holst, T. L., Kaynak, U., Gundy, K. L., Thomas, S. D., Flores, J., and Chaderjian, N. M. "Numerical Solution of Transonic Wing Flows Using an Euler/Navier-Stokes Zonal Approach." AIAA Paper No. 85-1640, July 1985.

8. Atta, E. H. and Vadyak, J. A. "A Grid Interfacing Zonal Algorithm for Three-Dimensional Transonic Flows About Aircraft Configurations." AIAA Paper No. 82-1017, June 1982.

9. Benek, J. A., Steger, J. L., and Dougherty, F. C. "A Flexible Grid Embedding Technique with Application to Euler Equations." AIAA Paper No. 83-1944, July 1983.

10. Venkatapathy, E. and Lombard, C. K. "Flow Structure Capturing on Overset Patch Meshes." AIAA Paper No. 85-1690, July 1985.

11. Berger, M. J. "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations." Stanford University STAN-CS-82-924, August 1982.

12. Buning, P. G. and Steger, J. L. "Graphics and Flow Visualization in Computational Fluid Dynamics." AIAA Paper No. 85-1507, July 1985.

13. Benek, J. A., Buning, P. G., and Steger, J. L. "A 3-D Chimera Grid Embedding Technique." AIAA Paper No. 85-1523, July 1985.

14. Benek, J. A., Steger, J. L., Dougherty, F. C., and Buning, P. G. "Chimera: A Grid Embedding Technique." AEDC-TR-85-64 (AD-A167466), April 1986.

15. Baldwin, B. S. and Lomax, H. "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows." AIAA Paper No. 78-257, January 1978.

16. Beam, R. and Warming, R. F. "An Implicit Finite Difference Algorithm for Hyperbolic Systems in Conservation Law Form." Journal of Computational Physics, Vol. 22, No. 1, September 1976, pp. 87-110.

17. Beam, R. and Warming, R. F. "An Implicit Factored Scheme for Compressible Navier-Stokes Equations." AIAA Paper No. 77-645, June 1977.

18. Pulliam, T. H. and Steger, J. L. "On Implicit Finite Difference Simulations of Three-Dimensional Flows." AIAA Paper No. 78-10, January 1978.

19. Kraft, E. M., Ritter, A., and Laster, M. L. "Advances at AEDC in Treating Transonic Wind Tunnel Wall Interference." Presented at the 15th Congress, International Council of the Aeronautical Sciences, London, UK, September 1986.

20. Suhs, N. E. "Computational Estimates of Strut Support Interference at Transonic Mach Numbers." AIAA Paper No. 85-5018, October 1985.

21. Sorenson, R. L. "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by Use of Poisson's Equations." NASA TM 81198, May 1980.

22. Soni, B. K. "Two and Three-Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics." AIAA Paper No. 85-1526, July 1985.

23. Thompson, J. F. "A Composite Grid Generation Code for General 3-D Regions." AIAA Paper No. 87-275, January 1987.

24. Nenni, J. P., Erickson, J. C., Jr., and Wittliff, C. E. "Measurement of Small Normal Velocity Components in Subsonic Flows by Use of a Static Pipe." AIAA Journal, Vol. 20, No. 8, August 1982, pp. 1077-1083.

25. Jacocks, J. L. "Aerodynamic Characteristics of Perforated Walls for Transonic Wind Tunnels." AEDC-TR-77-61 (AD-A040904), June 1977.

26. Donegan, T. L., Benek, J. A., and Erickson, J. C., Jr. "Calculation of Transonic Wall Interference." AIAA Paper 87-1432, June 1987.

27. King, H. A. and Sneden, K. J. "Weapon Integration: Key to the 'Clean Machine'." Aerospace America, Vol. 22, No. 8, August 1984, pp. 66-68.

28. Kaufman, L. G. II, Maciulaitis, A., and Clark, R. L. "Mach 0.6 to 3.0 Flows Over Rectangular Cavities." Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, AFWAL-TR-82-3112, May 1983.

29. Rossiter, J. E. "Wind Tunnel Experiments on the Flow Over Rectangular Cavities at Subsonic and Transonic Speeds." British A.R.C., R & M No. 3438, 1966.

30. McGregor, O. W. and White, R. A. "Drag of Rectangular Cavities in Supersonic and Transonic Flow Including the Effects of Cavity Resonance." AIAA Journal, Vol. 8, No. 11, November 1970, pp. 1959-1964.

31. Suhs, N. E. "Computations of Three-Dimensional Cavity Flow at Subsonic and Supersonic Mach Numbers." AIAA 87-1208, June 1987.

32. Cottrell, C. J., Martinez, A., and Chapman, G. T. "A Study of Multi-Body Aerodynamic Interference at Transonic Mach Numbers." AIAA Paper No. 87-0519, January 1987.

33. Degani, D. and Schiff, L. B. "Computation of Supersonic Viscous Flows Around Pointed Bodies at Large Incidence." AIAA Paper No. 83-0034, January 1983.

34. Kaynak, J., Cantwell, B. J., Holst, T. L., and Sorenson, R. L. "Numerical Simulation of Transonic Separated Flows Over Low Aspect Ratio Wings." AIAA Paper No. 86-0508, January 1986.

35. Benek, J. A., Donegan, T. L., and Suhs, N. E. "Extended Chimera Grid Embedding Scheme with Application to Viscous Flows." AIAA Paper No. 87-1136-CP, June 1987.

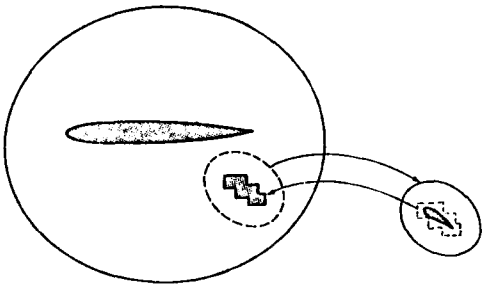


Fig. 1. Transfer of Information Between the Grids.

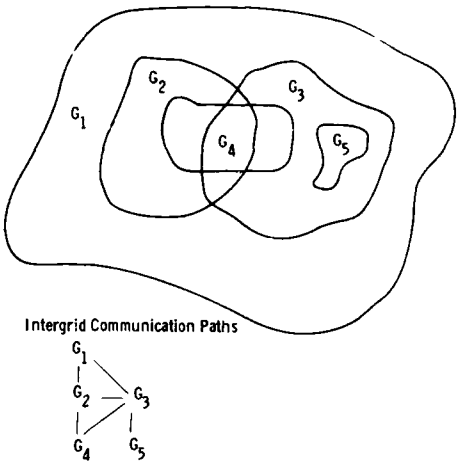


Fig. 2. Structure of Embedded Grids.

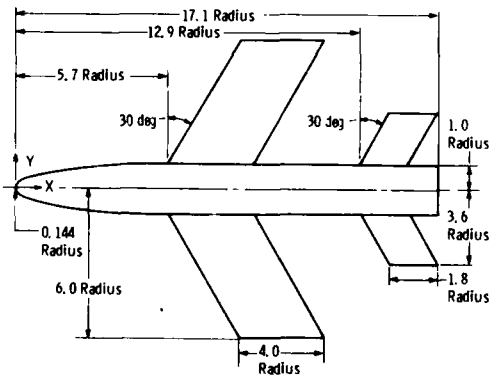


Fig. 3. Wing/Body/Tail Configuration.

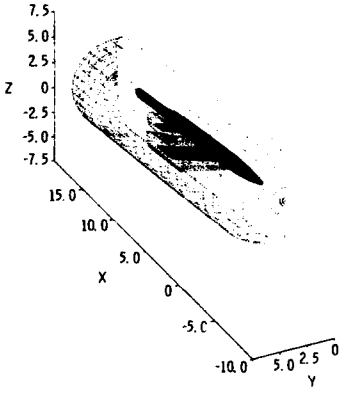


Fig. 4. Composite Grid for Fuselage, Sting, Wing, and Tail Grids.

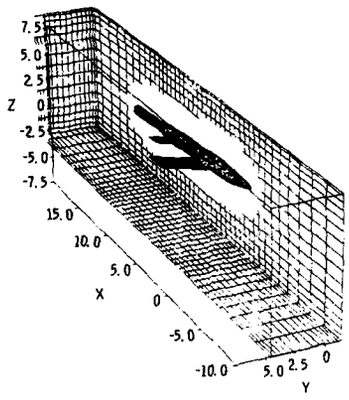


Fig. 5. Model Grids Embedded in the Tunnel Grid.

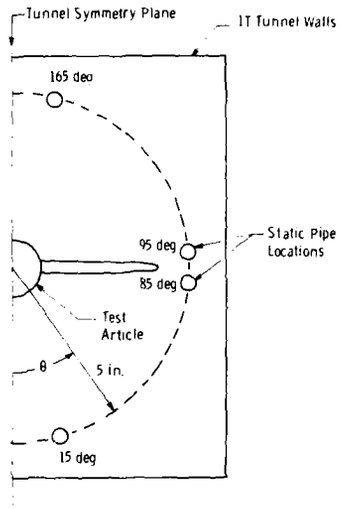


Fig. 6. Schematic of Mobile Static Pipe Locations in Tunnel 1T.

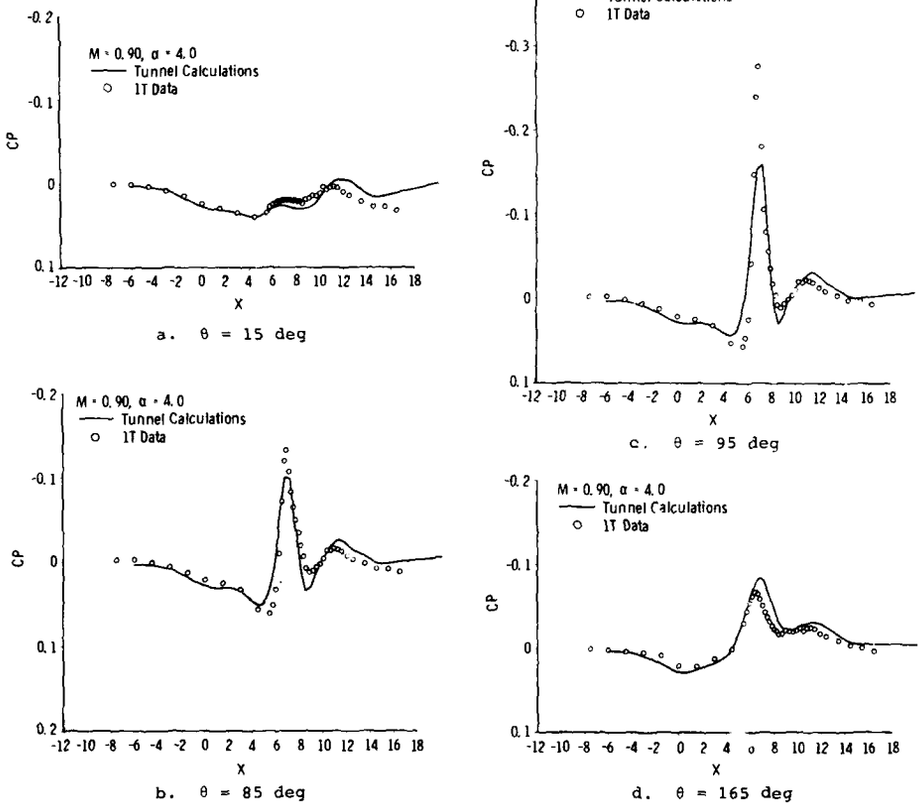
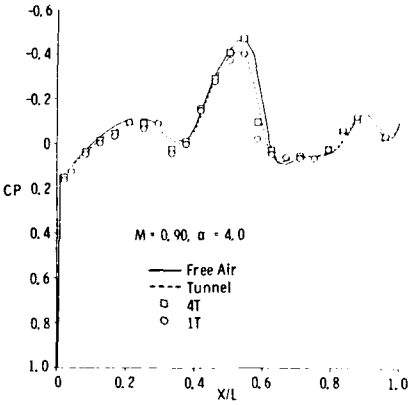
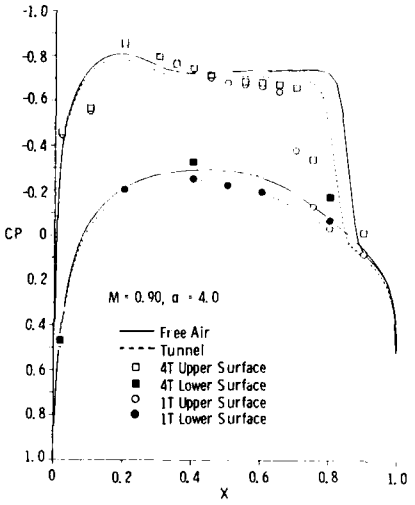


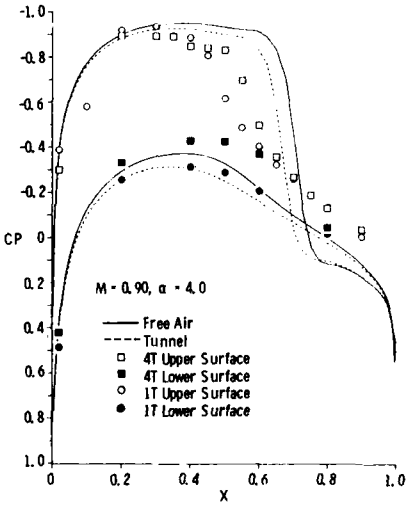
Fig. 7. Comparisons of Measured Static Pipe Pressure Data with Wing/Body/Tail Tunnel Calculations.



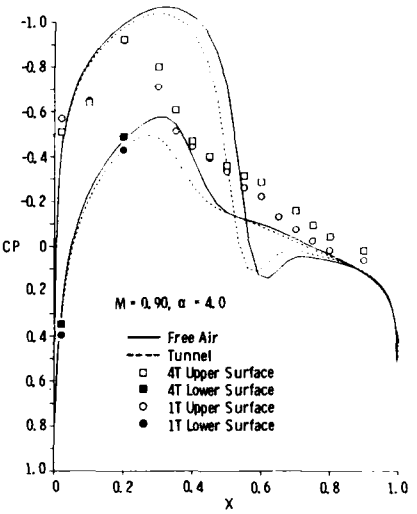
a. Crest of Fuselage



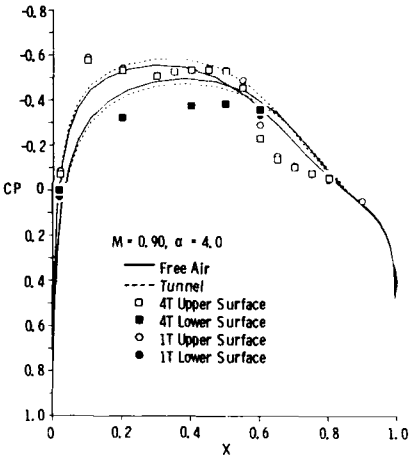
b. Wing Span 40 Percent



c. Wing Span 60 Percent



d. Wing Span 90 Percent



e. Tail Span 60 Percent

Fig. 8. Comparisons of Measured Static Pressure Data and Computations on Model Surface.

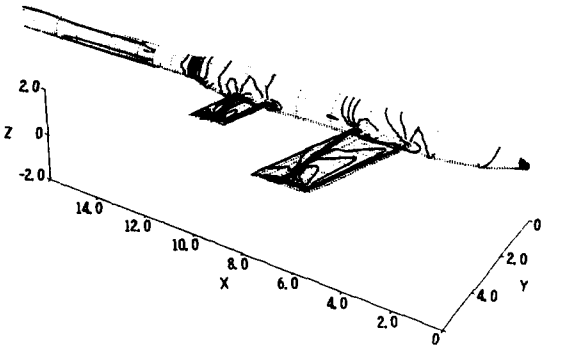


Fig. 9. Mach Number Contours for Tunnel Solution of Wing/Body/Tail Configuration, $M_\infty = 0.9$, $\alpha = 4$ Deg.

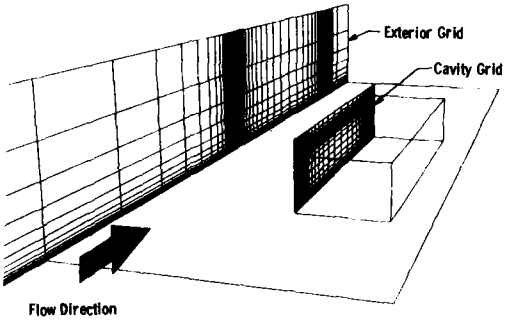


Fig. 10. Computational Grids.

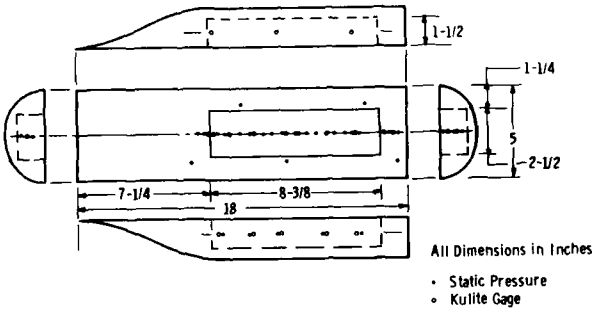


Fig. 11. Model Instrumentation Locations.

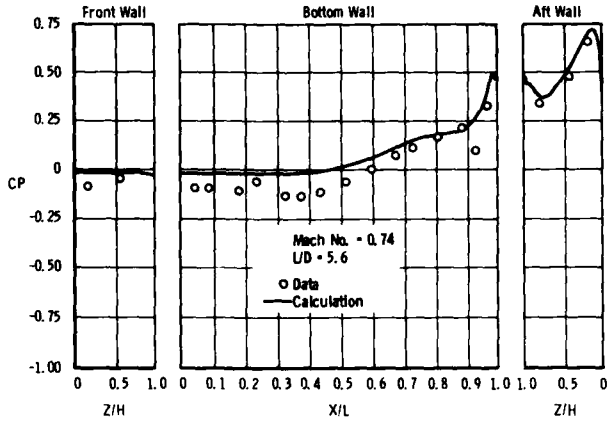


Fig. 12. Pressure Coefficient (CP).

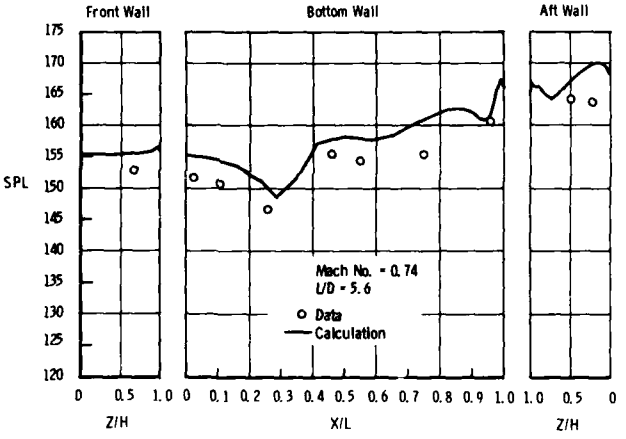


Fig. 13. Sound Pressure Level (SPL).

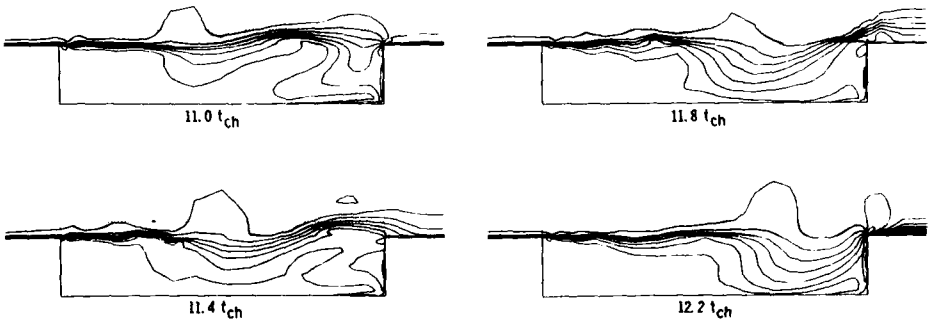


Fig. 14. Mach Contours - Cavity Plane of Symmetry (Mach 0.74).

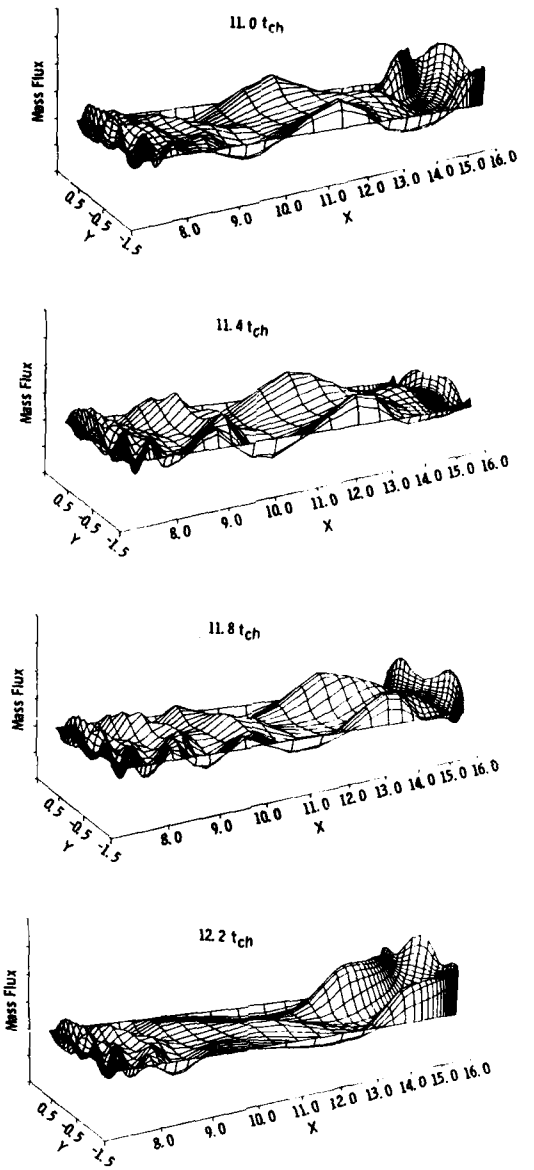


Fig. 15. Mass Flux Across Cavity Opening (Mach 0.74).

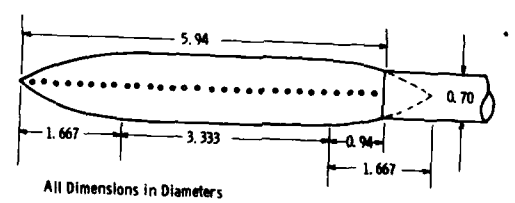


Fig. 16. Single Body of Revolution.

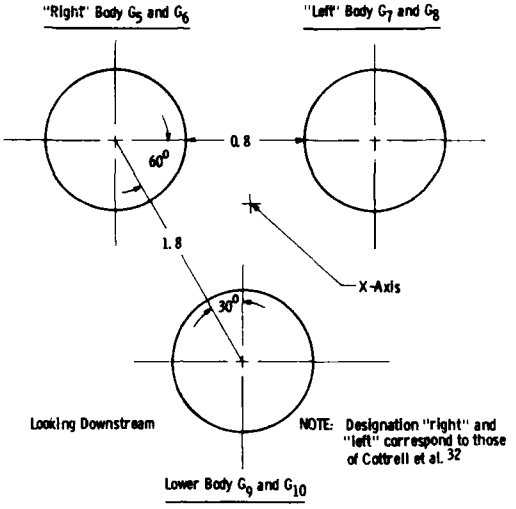


Fig. 17. Arrangement of Three Body Configuration Looking Downstream. Dimensions in Body Diameters.

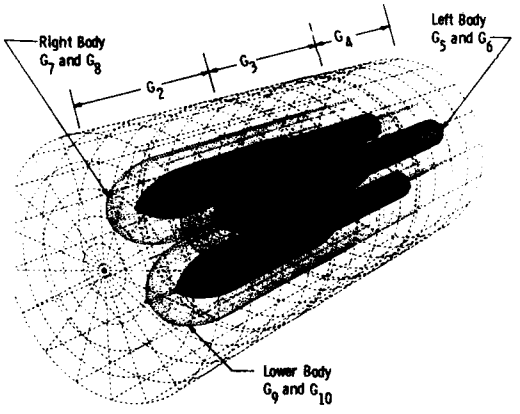


Fig. 18. Arrangement of Component Grids.

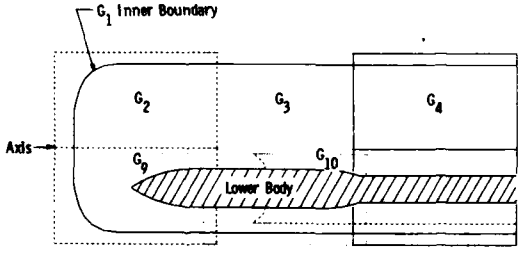
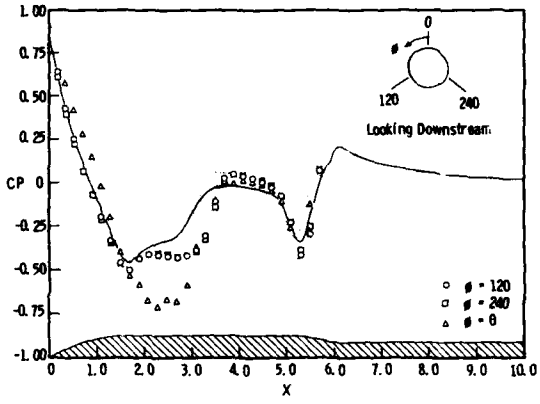
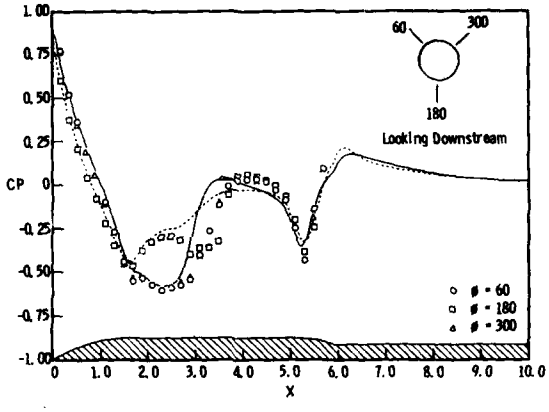


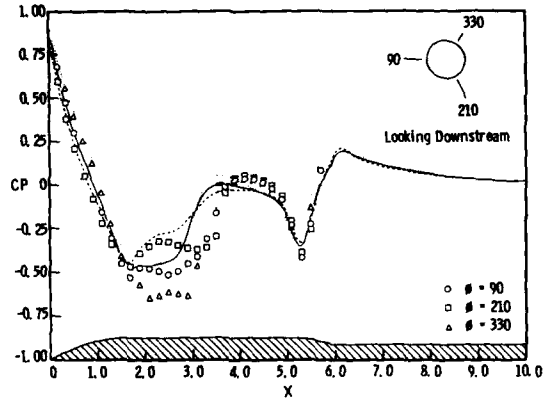
Fig. 19. Cross Section of Composite Mesh Through Symmetry Plane of Lower Body.



a. Azimuthal Locations - ϕ = 0, 120, and 240 Deg

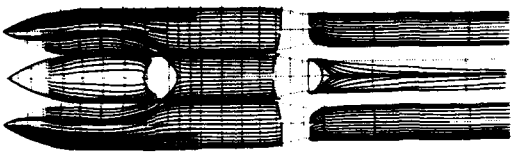


b. Azimuthal Locations - ϕ = 60, 180, and 300 Deg



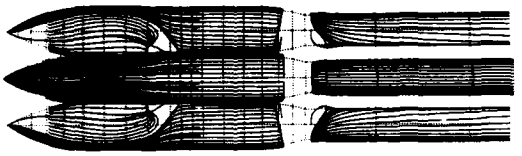
c. Azimuthal Locations - ϕ = 90, 210, and 330 Deg

Fig. 20. Experimental and Computed Axial Distributions of Pressure Coefficient for M_∞ = 0.95, α = 0 for Lower Body.



a. Top View Computation

b. Top View Experiment



c. Bottom View Computation

d. Bottom View Experiment

Fig. 21. Comparison of Computed and Measured Oil Flows for $M_\infty = 0.95$, $\alpha = 0$.

4.13 **GRID GENERATION AROUND TRANSPORT AIRCRAFT CONFIGURATIONS USING A MULTI-BLOCK STRUCTURED COMPUTATIONAL DOMAIN**

R. Radespiel

Deutsche Forschungs- und Versuchsanstalt für
Luft- und Raumfahrt e.V.
Institut für Entwurfsaerodynamik
D-3300 Braunschweig-Flughafen
F.R. Germany

SUMMARY

A new grid generation code is described which is based on the multi-block approch. Grid generation around three-dimensional configurations is divided into three major parts, namely surface definition, surface grid generation and field grid generation. Coons' patches are used to define the surfaces and their intersection lines. Surface grids and field grids are generated using the numerical solution of an elliptic system. An effective means for the control of the grid spacing has been developed which is based on an iterative determination of the source terms in the elliptic system. The code is used to generate grids around a wing-body combination and a high bypass nacelle configuration.

1. INTRODUCTION

Many codes for the computation of three-dimensional flows use rather simple grid generation procedures, such as stacking two-dimensional grids. These codes cannot be expected to resolve accurately regions of three-dimensional geometry, for example the vicinity of the wing tip or complex intersections of aerodynamic surfaces.

Essentially three-dimensional generation methods can suffer from problems due to overlapping of grid lines in the physical domain and may generate grids with discontinuities of the metrics in the field as often occurs in the case of algebraic generation systems. Codes based on differential equation systems require much more computing time and often there are no means available for a direct control of the grid density everywhere.

We postulate that the first requirement of a 'good' grid generation method should be user friendliness. The user should be able to cluster grid points in regions where he expects the flow gradients to be high. The code should be robust, which means it should be applicable to a wide range of configurations without changing the numerical parameters of the method. Secondly the code should be at least one order of magnitude faster than the flow solver under consideration. We believe no grid generation code would ever give the optimal distribution of grid points with its first run. After visual inspection of the grid the user will decide either to change the input of the grid generation code and makes a new run of the program or to use the grid for flow computations. Thirdly the part of the computer program, which depends on the specific configuration under consideration should be as small as possible. This requirement leads to the concept of generating block structured grids as used in [1, 2, 3, 4].

The present report describes recent work to meet these requirements. First the problem of grid topologies around aircraft configurations is discussed in some detail. In section 3 the surface equations which are used to define arbitrary, intersecting surfaces are given according to Coons [5]. The generation of both two-dimensional surface grids and three-dimensional field grids is based on the numerical solution of an elliptic system as outlined in [6]. An effective means for the control of the grid point distribution in the field has been developed which uses an iterative determination of the source terms in the elliptic system. This procedure will be described in section 4. The computer program is based on the multi-block approach. Therefore, the mayor part of the program will be independent of the problem under consideration. An outline of the computational procedure will be discussed in section 5. Finally, the results of the generation of grids around a wing-body and a high bypass nacelle configuration are given.

2. GRID TOPOLOGIES FOR TRANSPORT AIRCRAFT CONFIGURATIONS

It is well known that there does not exist an optimal grid topology for arbitrary aircraft configurations. Each aerodynamic component of an aircraft may have its own natural grid structure and usually these natural structures of the components can not be patched with each other. For a given configuration one has first to decide about the global grid topology. The global grid should be compatible with all local subgrids, which are used for resolution of the individual components.

For the particular configuration of a transonic transport aircraft the following main components have to be considered: Large aspect ratio, moderately swept wing, blunt fuselage, moderately swept empennage, engine nacelle mounted on strut. If all these components have to be integrated into the grid an H-type sectionwise global grid seems appropriate. For H-sections one family of grid lines will approximately follow the streamlines so that the lifting surfaces can be represented as interior slits. In the spanwise

direction the grid lines should enclose the wing forming an "O". Compared with an H-type spanwise grid, an O-grid will save 25-30% of the grid points and thus computer time. In conclusion, the global grid now has an H-O-structure with respect to the wing. The additional components of the aircraft can be integrated into the mesh as follows:

- The fuselage can be mapped directly into the global grid. The grid structure with respect to the fuselage is then H-H. Another possibility is to embed a local O-O grid around the fuselage into the global grid. This grid arrangement is sketched in Figure 1. Compared to the first alternative the arrangement with the embedded subgrid will save 40% of the grid points when generating grids for a numerical solution of the Euler equations. For a numerical solution of the Navier Stokes equations the total number of grid points may be reduced by 60%.

- The horizontal tailplane can be represented as an interior slit in both the global H-O grid and the subgrid of the fuselage. If the resolution of these grids is not good enough in the region of the tailplane a local O-O subgrid can be embedded into the slit. The same procedure can be followed in the case of the vertical tailplane.

- For the integration of the engine nacelle, a portion of the global grid below the wing can be replaced by a polar subgrid, whose singularity is on the axis of the nacelle. The resolution of the strut may give problems if it is highly swept. In this case it should be advantageous to use additional singular grid points on the strut surface to avoid highly skewed grid lines on it.

3. SURFACE DEFINITION

One of the basic requirements of a grid generator is to allow a complete mathematical description of all the aerodynamic surfaces under consideration. In practice, aerodynamic surfaces are given by using cross sections; i.e., a fuselage is described by cross sections along the body axis and a wing is given by airfoil sections in the spanwise direction. The first step of the computational procedure is thus the generation of a mathematical description of the surfaces which fits the input cross sections and provides at least continuous derivatives up to first order. The description should contain the intersection lines of the aerodynamic surfaces. A convenient method of surface definition which meets these requirements uses Coons'patches [5].

Using the input cross sections any surface can be covered with a grid of patches. The distribution of the cartesian coordinates over a single patch is considered now. For this purpose two independent coordinates u, $0 \leq u \leq 1$ and w, $0 \leq w \leq 1$ can be defined according to Figure 2. Following Coons a surface equation can be derived which gives both continuous coordinates and continuous slopes across the boundaries $u = 0$, $u = 1$, $w = 0$, $w = 1$. Defining the boundary curves of the patch as cubic polynomials the surface equation can be written as

$$\hat{x} = \hat{u}\, \bar{M}\, \hat{w}^T \qquad (1)$$

where $\hat{x} = [x(u,w),\ y(u,w),\ z(u,w)]$ contains the components of the cartesian coordinates,

$$\hat{u} = [u^3,\ u^2,\ u,\ 1] \quad ; \quad \hat{w} = [w^3,\ w^2,\ w,\ 1]$$

and $\bar{M}$ is a matrix containing the parametric derivatives of $\hat{x}$ combined with blending functions to provide continuity across the boundary curves. For each patch the elements of $\bar{M}$ only need to be calculated once and stored.

Once the patches of all the surfaces have been defined, intersections of any two surfaces can be calculated. A point $\hat{x}$ lying on the intersection line of two patches must satisfy equation (1) for each patch. Then, setting these equations equal, three non-linear algebraic equations for the four parametric coordinates of the two patches are generated. If one of the parameters is fixed, i.e. percent line of the wing, the equations can be solved to give the corresponding point on the intersection line.

In conclusion, all the aerodynamic surfaces are continuously described by dividing them into a number of Coons' patches, which use two parametric surface coordinates u and w. Hence, a unique transformation $\hat{x} = \hat{x}(u, w)$ has been established for each patch. In addition to the possibility of calculating intersections between any two surfaces, this transformation can also be used for the generation of surface grids. This is described in section 4.

4. GRID GENERATION SYSTEM

At the present time the choice of a specific grid generation system seems to be based more on the engineer's intention and his particular experiences than on established theorems. In our case it is felt that an elliptic generation system offers the most flexibility to treat complex three-dimensional geometries. In particular, for H-topologies, is seems to be rather difficult to use algebraic generation equations to generate smooth grids without any overlap of grid lines.

For the numerical solution of an elliptic system in a three-dimensional computational domain, the definition of the grid point distribution at the boundaries of the domain is required. Hence, a three-dimensional field grid generation system and a two-dimensional surface grid generation system have to be specified. Surface grids must be generated at the farfield boundaries of the computational domain and on the surface of the aircraft configuration. The grids on the surface will in turn influence the field grid close to the surface, where high flow gradients are expected. Hence, the generation of proper surface grids is an important aspect of the total problem. In the following we will first introduce the generation system for three-dimensional fields and then discuss the surface grid generation system.

4.1 Generation System of Field Grids

Thompson et al [6] have given an elliptic generation system

$$\vec{\xi}_{xx} + \vec{\xi}_{yy} + \vec{\xi}_{zz} = \vec{P}\, J^2\, \bar{A} \tag{2}$$

for the curvilinear coodinates $\vec{\xi} = [\xi, \eta, \zeta]^T$. Here J^2 denotes the square of the Jacobian and the elements of $\bar{A} = [^{A}B_{C}]$ are functions of the transformation coefficients. The Laplace operator on the left hand side of system (2) will provide a smooth distribution of the coordinates $\vec{\xi}$ in physical space. Furthermore, it has beeen shown [7], that the system (2) exhibits an extremum principle, if the inhomogenuous functions $\vec{P} = [P, Q, R]^T$ vanish, i.e. the generation system then guarantees a one-to-one mapping for boundary-conforming curvilinear systems on general closed boundaries. However, this condition with respect to $\vec{P}$ is not necessary to generate non-overlapping grids and, in practice, the source terms $\vec{P}$ which are used to control spacing and orientation of grid line will have to be large and change their sign within the field in order to generate suitable grids. The determination of these terms will be discussed in section 4.3.

In order to obtain the location vector $\vec{x} = f(\vec{\xi})$ it is convenient to interchange the role of dependent and independent variables in equation (2), which gives a quasilinear elliptic system for the cartesian position vector $\vec{x}$:

$$A(\vec{x}_{\xi\xi} + P\vec{x}_{\xi}) + B(\vec{x}_{\eta\eta} + Q\vec{x}_{\eta}) + C(\vec{x}_{\zeta\zeta} + R\vec{x}_{\zeta}) + 2(D\vec{x}_{\xi\eta} + E\vec{x}_{\xi\zeta} + F\vec{x}_{\eta\zeta}) = 0 \tag{3}$$

The coefficients A to F are related to the transformation coefficients $\vec{x}_{\xi}$, $\vec{x}_{\eta}$ and $\vec{x}_{\zeta}$. Equation (3) can be solved in the computational domain to yield the location vector $\vec{x}$ at each discrete value of $\vec{\xi}$. For this purpose a conventional successive line relaxation method is currently used.

4.2 Generation System of Surface Grids

Surface grids are generated using two parametric coordinates. In the most simple case the surface lies in a plane in physical space, i.e. z = const. In this case the other two cartesian coordinates x and y can be used as parametric coordinates. A typical application of this is the generation of a surface grid in the symmetry plane of a wing. If a grid has to be generated on the surface of an aircraft configuration it is clear that the parametric coordinates u and w of the Coons' patches from section 3 can be used. For example, the parametric coordinates u and w on a fuselage may be defined as sketched in Figure 3a. The problem is now, the generation new curvilinear coordinates ξ, η on this surface (Figure 3b) which conform to the intersection between fuselage and wing.

Thompson et al [6] have derived a general elliptic surface grid generation system which takes into account the partial derivatives of the transformation $\vec{x} = \vec{f}(u, w)$, relating to curvature, skewness or stretching of the Coons' patches. Using this generation system the surface grid will be virtually independent of the original definition of the Coons' patches. In our particular case, generating a surface grid on an aircraft fuselage, there are no problems associated with the metrics of the parametric coordinates. On most of the fuselage surface the new curvilinear coordinates will follow the parametric coordinates closely. Furthermore, the coordinates u and w of the Coons' patches and the new curvilinear coordinates $\vec{\xi} = [\xi, \eta]^T$ will both have two singular points, one at the nose of the fuselage and one at the tail.

In the present work the surface grids are therefore generated using simply the two-dimensional version of system (2):

$$\begin{aligned} \xi_{uu} + \xi_{ww} &= P\, J^2\, A \\ \eta_{uu} + \eta_{ww} &= Q\, J^2\, B \end{aligned} \tag{4}$$

in the parametric u-w domain. The source terms P and Q are used to control the spacing of the surface grid. Their determination is given in section 4.3.

4.3 Iterative Grid Control

The source terms $\vec{P} = [P, Q, R]$ can be used to control spacing and orientation of the curvilinear coordinates $\vec{\xi}$ in physical space. A user of the grid generation code usually wants to cluster grid lines in regions where he expects the flow gradients to be high. For example he may wish to specify a certain grid spacing close to the aerodynamic surface. As an alternative the user may choose a particular plane in the computational domain for which a point distribution is to be prescribed. This certain plane may be a boundary of the computational domain or it may be an interior plane, at which a desired point distribution is to be specified. Thompson [6] has given a relatively simple estimate of the source terms in order to achieve a desired point distribution. For example, let η = const. be the plane under consideration. Assume that the coordinate line on which η varies crosses that plane orthogonally and that the curvature of the coordinate line on which η varies is zero. The source terms P and R then may be written

$$P = -\frac{s_{\xi\xi}}{s_\xi} \quad , \qquad R = -\frac{s_{\zeta\zeta}}{s_\zeta} \tag{5}$$

where s denotes the arc length distribution along the coordinates ξ and ζ. Similar expressions can be given for source terms on planes ξ = const and ζ = const. Once the source terms have been evaluated on certain planes they can be interpolated in the whole region. Generally, the resulting grids will not exhibit the desired point distribution on the planes which were initially used to evaluate equation (9), because both skewness and curvature of the lines crossing those planes are not taken into account. The grid lines will move toward convex boundaries and they will move away from concave boundaries. This tendency is sketched in Figure 4 for the case of a C-grid around a highly cambered airfoil. Thompson et al [8] have given expressions which take into account the curvature of the boundaries of the computational domain. However, the curvature of the boundary will not necessarily reflect the curvature of the grid lines in the field. Furthermore it may be difficult to evaluate the curvature at boundaries of an H-type topology as shown in Figure 5.

As both curvature and skewness of the grid lines come out as a part of the solution and therefore cannot be calculated beforehand, an iterative determination of source terms has been developed. In order to obtain a desired point distribution on certain planes of the computational domain the source terms are adjusted throughout the whole solution process. From here on, these planes will be called target planes. On a target plane η = const target values of the grid stretching $(s_{\xi\xi}/s_\xi)_o$ and $(s_{\zeta\zeta}/s_\zeta)_o$ can be calculated from the desired point distribution on that plane. The iterative solution of the elliptic system (5) yields new values of the coordinates after each iteration n. From these new coordinates actual values of $(s_{\xi\xi}/s_\xi)_n$ can be computed. The difference between target values and actual values of the grid stretching can be used to adjust the source terms on the target plane as follows:

$$P_{n+1} = P_n + c_p \left[\left[\frac{s_{\xi\xi}}{s_\xi} \right]_n - \left[\frac{s_{\xi\xi}}{s_\xi} \right]_o \right] \tag{6}$$

To obtain a stable iteration scheme it was found necessary to add a damping term such as the derivative of the difference between target values and actual values with respect to the iteration number. The final iteration formula then reads:

$$P_{n+1} = P_n + c_p \left[\left[\frac{s_{\xi\xi}}{s_\xi} \right]_n - \left[\frac{s_{\xi\xi}}{s_\xi} \right]_o \right] + c_T \left[\left[\frac{s_{\xi\xi}}{s_\xi} \right]_n - \left[\frac{s_{\xi\xi}}{s_\xi} \right]_{n-1} \right] \tag{7}$$

Analogous expressions follow for the iterative determination of R. Once the source terms P and R have been obtained on the target plane, they can be interpolated in the entire domain and the solution algorithm can proceed to the next iteration. The converged solution will yield a coordinate grid for which $(s_{\xi\xi}/s_\xi) = (s_{\xi\xi}/s_\xi)_o$ is valid on the target plane and which is smooth in the entire domain. In the case of generation of two-dimensional surface grids, the target planes reduce to target lines on which one source term can be evaluated from the arc length distribution.

Values of the coefficients $c_p = 0.1$ and $c_T = 0.2$ have been determined empirically. It has been found that the convergence behavior of the numerical solution with iterative grid control is not very sensitive with respect to c_p and c_T.

5. MULTI-BLOCK STRUCTURED COMPUTATIONAL PROCEDURE

In order to enable the treatment of complex configurations a multi-block structured grid generation code has been developed. In the multi-block approach a complicated multiply connected computational domain is split into a number of simply connected cubes. The solution algorithm is completely independent of the specific configuration under consideration. The definition of the information which is necessary to describe a general block structured domain has been fixed in cooperation with DORNIER Company and SUPRENUM Company and has already become a software standard [8]. This definition of the block structured domain is given in the following subsection. Afterwards a brief outline of the computer program is given. This subsection is illustrated by the results for a typical wing-body combination.

5.1 Definition of Block Structured Domains

A single block is formed by a right-hand system of the computational coordinates i=1,IL, j=1,JL, k=1,KL. The six block boundaries are numbered according to Figure 7a. The connection of blocks to form general regions is completely arbitrary and does not depend on the numbering of the blocks. The information necessary to define general regions is stored as follows:

- Each block boundary can be divided into an arbitrary number of segments, which either correspond to a segment of a neighbor block (inner cut) or are a part of the outer boundary of the computational domain. The dimension of the segments is fixed by a regular, non-equidistant segment grid on each block boundary. The segment grid is defined by an integer number IJKL for each block boundary

  IJ Number of segments in first cyclic direction at the block boundary

  KL Number of segments in second cyclic direction at the block boundary

  and an integer array MNOPQR (0:ID) with ID $\leq$ MAX(IJ, KL) for each block boundary

  MNO(I) Value of block coordinates in first cyclic direction at the segment grid lines.
  PQR(I) Value of block coordinates in second cyclic direction at the segment grid lines.

  The use of IJKL and MNOPQR is illustrated in Figure 7b.

- The neighbors of the blocks are defined using the integer numbers IJKLMN (ID, ID) and OPQR (ID, ID) for each block boundary. IJKLMN is coded for each segment as follows:

  IJK Number of the neighbor block. For IJK = 0 the segment is a part of the boundary of the computational domain.

  L Number of the neighboring block boundary

  MN Definition of the index orientation of the neighboring segment according to Figure 7.

  The integer array OPQR is coded for each segment as follows:

  OP Number of the neighboring segment in first cyclic direction of the segment grid.

  QR Number of the neighboring segment in second cyclic direction of the segment grid.

Using the integers IJKL, MNOPQR,IJKLMN, OPQR arbritrarily connected block structures are completely described.

5.2 Outline of the Grid Generation Code

The present grid generation code can be divided into three major parts, namely surface definition, surface grid generation and field grid generation. The results of each of the three mayor parts of the code are stored in a file. Therefore, the code can be run step by step, which allows the user to check the results of each step before proceeding to the next.

5.2.1 Surface Definition

The starting-point for the definition of the aerodynamic surfaces is the input of discrete surface sections. These input sections are shown for a typical transport configuration in Figure 8a. Cubic splines over arc length distributions are used to interpolate a smooth grid of Coons patches on the fuselage. On the wing cubic splines in the sectionwise direction and linear interpolation between the input sections in the spanwise direction are used. A smooth and closed grid around the wing tip is generated. For this purpose a series of superellipses is applied to the planform and the thickness of the wing in the region near the tip. The grids of Coons' patches for both the wing and the fuselage are shown in Figure 8b. The elements of $\bar{M}$ are calculated according to equation (1) and stored for all the patches. Finally the values of u and w at the intersection of wing and fuselage are calculated for fixed percent lines of the wing grid (see section 3). For each intersection point the system of three non-linear equations is solved by a library secant method for simultaneous non-linear equations [9]. It was found to give rapid convergence for all the configurations analysed to this point.

5.2.2 Surface Grid Generation

In the present version of the program the surface grid generation step contains those parts of the code which depend on the particular configuration under consideration. In the future, however, it is intended to generate the surface grids interactively on a work station. The choice of an appropriate grid topology for wing-body combinations has been discussed in section 2. Obviously surface grids have to be generated on the wing, the body, the farfield boundaries, in the symmetry plane of the grid and on all the fictitious inner cuts between the blocks. There are upper and lower blocks for the global H-O grid around the wing and upper and lower blocks around the fuselage. Note that generation of surface grids on fictitious inner cuts is necessary only to provide an initial grid for the field grid generation step. However, sometimes it may be useful to fix the coordinates at inner cuts. (See section 5.2.3).

Initially, algebraic grids are generated for all boundary segments of the three-dimensional domain. For any of these segments, the surface grid can be improved by solving the elliptic system (4). In this case a two-dimensional counterpart to the logic for the field blocks (see section 5.1) is provided automatically from the block structure of the field grids. To control the grid spacing at the boundary segments, target lines may be specified for each two-dimensional surface block (see section 4.3). The present code provides two options to determine target values of $(s_{\xi\xi}/s_{\xi})_o$ along a coordinate line on which i varies:

- Option INITIAL DISTRIBUTION
  $(s_{\xi\xi}/s_{\xi})_o$ is calculated from the arc length distribution of the algebraic grid

$$= 2\,\frac{s_{i-1} - 2s_i + s_{i+1}}{s_{i+1} - s_{i-1}}$$

- Option GEOMETRIC PROGRESSION

  From a specified grid spacing at i=1, Δs and the distance $s_{IL} - s_1$ the ratio r of successive grid intervals is calculated, so that $s_{IL} - s_1 = \Delta s(r^{IL-1}-1)/(r-1)$. The finite difference approximation of $(s_{\xi\xi}/s_{\xi})_o$ is $2(r-1)/(r+1)$. It is constant along i.

Now the solutions of the two-dimensional elliptic system can be calculated. All the 2-D grids are stored end-to-end in singly- dimensioned arrays in the main program. The corresponding arrays of the subroutines are multiply-dimensioned with their size and starting location passed through COMMON and arguement lists. Therefore, no I/O-work is required when the algorithm passes from one block to the next. The improvements of grid quality which can be obtained using iterative grid control are displayed in Figure 9. Figure 9a shows the surface grid of a fuselage without any source terms. A surface grid with the sources determined according to equation (5) is shown in Figure 9b. Iterative grid control has been used for the surface grid of Figure 9c. The resolution of this grid is much better in the leading edge and trailing edge regions where large gradients of the flow are expected.

In the present example of a transport configuration elliptic grid generation is performed on the fuselage surface, the symmetry plane and the upstream and downstream farfield boundaries.

5.2.3 Field Grid Generation

Once the block-structured domain has been defined and the surface grids have been generated and stored (see section 5.2.2) grid generation proceeds in a manner independent of the particular configuration.

To initiate the numerical solution of the three-dimensional elliptic system all the surface grids of the boundary segments are collected from file. Then trilinear transfinite Lagrange interpolation is used to generate the initial grids in the interior of the blocks. To control the grid spacing, target planes may be chosen for each three-dimensional field block (see section 4.3). On each of these target planes two source terms may be specified which are determined from the distribution of arc length in the two coordinate directions of the plane. For example, on a plane on which j = const the source terms P and R are determined from the arc length distributions in the i- and k-directions, respectively. The present code provides three options to determine the target values of the source terms. In the case of the source term P there are:

- Option INITIAL DISTRIBUTION PLANE
  $(s_{\xi\xi}/s_{\xi})_o$ is calculated from the arc length distribution of the initial grid

$$= 2\,\frac{s_{i-1} - 2s_i + s_{i+1}}{s_{i+1} - s_{i-1}}$$

- Option INITIAL DISTRIBUTION BOUNDARY
  $(s_{\xi\xi}/s_{\xi})_o$ is calculated from the arc length distribution of the initial grid at k=1 and k=kL and is linearily interpolated for intermediate values of k.

- Option GEOMETRIC PROGRESSION
  From a specified grid spacing at i=1, Δs and the distance $s_{IL} - s_1$ the ratio of successive grid intervals r is calculated, so that $s_{IL} - s_1 = \Delta s(r^{IL-1} - 1)/(r-1)$. The finite difference approximation of $(s_{\xi\xi}/s_\xi)_0$ is $2(r - 1)/(r + 1)$.

When specifying the source terms P, Q and R care must be taken that the sources are not overspecified. Each source term must be specified using one family of target planes only. For example, the sources P may not be specified using both target planes i=const and j=const. Furthermore, care should be taken that the specified target values of the source terms are physical, which means one can expect a numerical solution of equation (3) to exist which fulfills the target values of the arc length distributions without excessive skewness or cross-over of grid lines.

Now the solutions of the three-dimensional elliptic system can be calculated. As in the surface generation step singly-dimensioned arrays in the main program and multiply-dimensioned arrays in subroutines are used to store the grids. However, due to the limited amount of main memory of the Cray 1-S computer used in the present work, only one field block is kept in the main memory at a time. At each iteration step of the solution algorithm each block is read from disk (BUFFER-IN), the iteration is executed and the block is written back to disk (BUFFER-OUT). The Record-Addressable READMS/WRITMS package is used for the exchange of the grid points on fictitious inner cuts. The ratio between I/O and CPU-time is about 3.

In order to reduce the computational expense of the grid generation method, successive grid refinement can be used in the field grid generation step. The solution of the coarse grid is interpolated and used as input for the refined grid calculation. Using this strategy, the iterative determination of the source terms, which can require several hundred iterations, can be done mainly on the coarsest grid. The task of the iterations on the finer grids is to smooth the intial interpolated grids, which only requires several dozen iterations. The convergence of the field grid generation method may be affected by the problem of cross-over of grid lines near singular points or lines of the grid. The H-O-topology of the present global grid shows two parabolic singular lines emanating from the wing tip. There is a natural tendency of the grid lines running around the singular lines to move very close to these lines. Although the continuum equations (2) possess a maximum principle under certain conditions this property does not apply to the discrete equations. Furthermore the initial solution with respect to both $\vec{x}$ and $\vec{P}$ may be so far away from the final converged solution, that cross-over of grid lines without recovery occurs near the singular lines even if a converged solution without cross-over does exist. The cross-over problem has also been addressed by Weatherill et al [10]. In the present work converged solutions for H-O-topologies have been obtained by

- generating a smooth surface grid on the last spanwise plane k=KL in the surface grid generation step and then fixing the coordinates on this plane in the field grid generation step,

- solving the elliptic system first with the source terms in quasi-spanwise direction R=0 for some hundred iterations on the coarse grid and then introducing the complete iterative grid control.

Clearly, the problem of grid cross-over near singular lines has not been solved satisfactorily with the present work and will require future effort.

To demonstrate the capabilities of the present method, field grids around a typical wing-body combination have been generated using 92 x 40 x 12 cells in the global H-O grid around the wing and 76 x 24 x 4 cells in the local O-O grid around the body. The numerical solution of the elliptic system was obtained using one grid refinement. Several views of sections of the final grid are shown in Figure 10. Both the wing and the fuselage are well resolved.

As a second application a grid around an isolated axisymmetric nacelle has been generated using the two dimensional part of the grid generation code only. In this case three blocks have been used to form an H-grid sectionwise and a polar structure in circumferential direction. Figure 11 shows, that all components of the configuration are well resolved.

6. CONCLUSIONS

A new grid generation code has been developed which is based on the multi-block approach. Grid generation around complex three- dimensional configurations is divided into three major parts, namely surface definition, surface grid generation and field grid generation. Surface definition is done using Coons' patches. Surface grids and field grids are generated using the numerical solution of an elliptic system. The elliptic system provides smooth grids in the interior of the computational domain, even if the distribution of grid points at the boundaries is not smooth. An effective means for the control of the grid spacing in the field has been developed which uses an iterative determination of the source terms in the elliptic system.

The code has been used to generate a grid around a wing-body combination which is typical for transport aircraft. As a further step towards the representation of complete transport aircraft, a grid around an isolated nacelle has been generated. The grids show a good resolution of the aerodynamic surfaces.

7. REFERENCES

[1] Lee, K.D.
Rubbert, P.E.
Transonic Flow Computations Using Grid Systems with Block Structure.
7th Int. Conference on Numerical Methods in Fluid Dynamics, pp. 266-271, (1980).

[2] Weatherill, N.P.
Forsey, C.R.
Grid Generation and Flow Calculation for Aircraft Geometries.
Journal of Aircraft, Vol. 22, pp. 855-860, (1985).

[3] Fritz, W.
Numerical Grid Generation Around Complete Aircraft Configurations.
AGARD-CP-412 (1986).

[4] Thompson, J.F.
A Composite Grid Generation Code for General 3-D Regions.
AIAA Paper 87-0275, (1987).

[5] Coons, S.A.
Surface for Computer-Aided Design of Space Forms.
MID MAC-TR-41, (1967).

[6] Thompson, J.F.
Warsi, Z.U.A.
Mastin, G.W.
Numerical Grid Generation.
North Holland, New York, (1986)

[7] Mastin, C.W.
Thompson, J.F.
Elliptic Systems and Numerical Transformations.
Journal. Math. App. 62, 52, (1978).

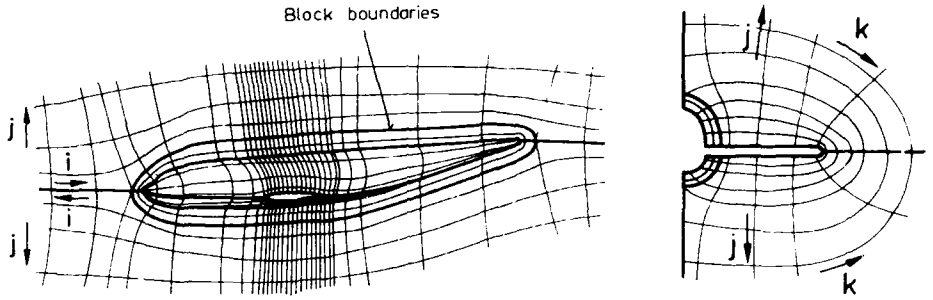
[8] SUPRENUM GmbH.
Spezification Band VI, Anwendungssoftware.
SUPRENUM Gesellschaft für numerische Superrechner mbH, Dezember 1986.

[9] IMSL
IMSL Library Reference Manual - Volume 4, Edition 9, (1982).

[10] Weatherill, N.P.
Shaw, J.A.
Forsey, C.R.
Rose, K.E.
A Discussion on Mesh Generation Technique Applicable to Complex Geometries.
AGARD-CP-412, (1986).

8. ACKNOWLEDGEMENT

Part of this work was supported by the Federal Ministry of Research and Technology under account ITR 8502I/4.



a) Grid in symmetry plane and on body surface b) Grid in spanwise section

Fig. 1: Grid topology for wing-body combination

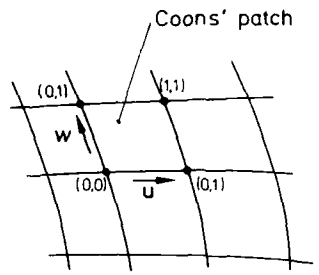
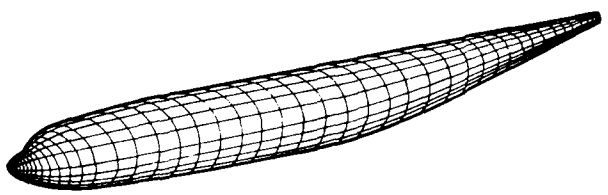
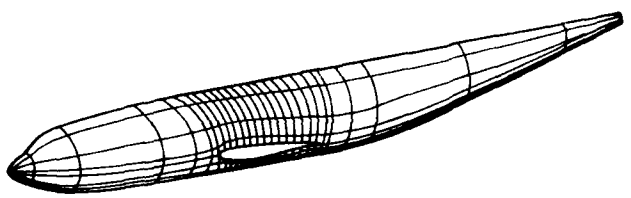


Fig. 2: Definition of Coons' patch



a) Distribution of the parametric coordinates u and w on a fuselage



b) Curvilinear surface coordinates ξ and ζ on a fuselage

Fig. 3: Surface grid generation

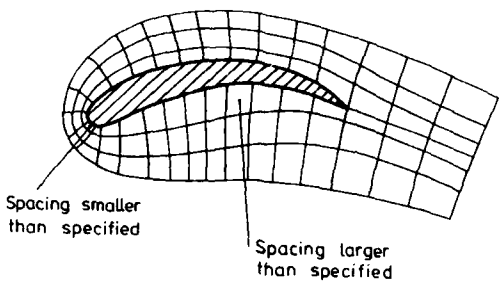


Fig. 4: Influence of surface curvature on grid spacing

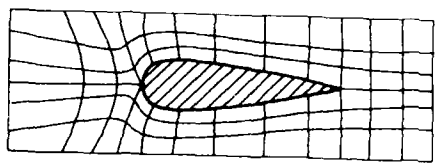
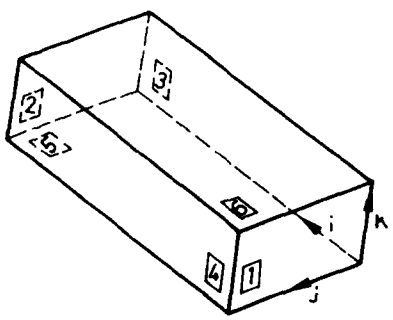
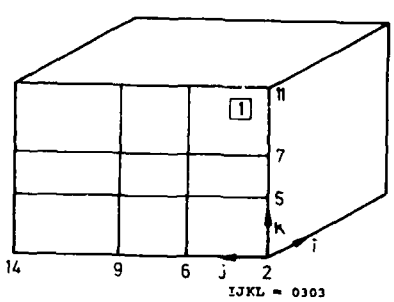


Fig. 5: H-type grid around airfoil showing discontinuity of slope at the leading edge



a) Numbering of the block boundaries



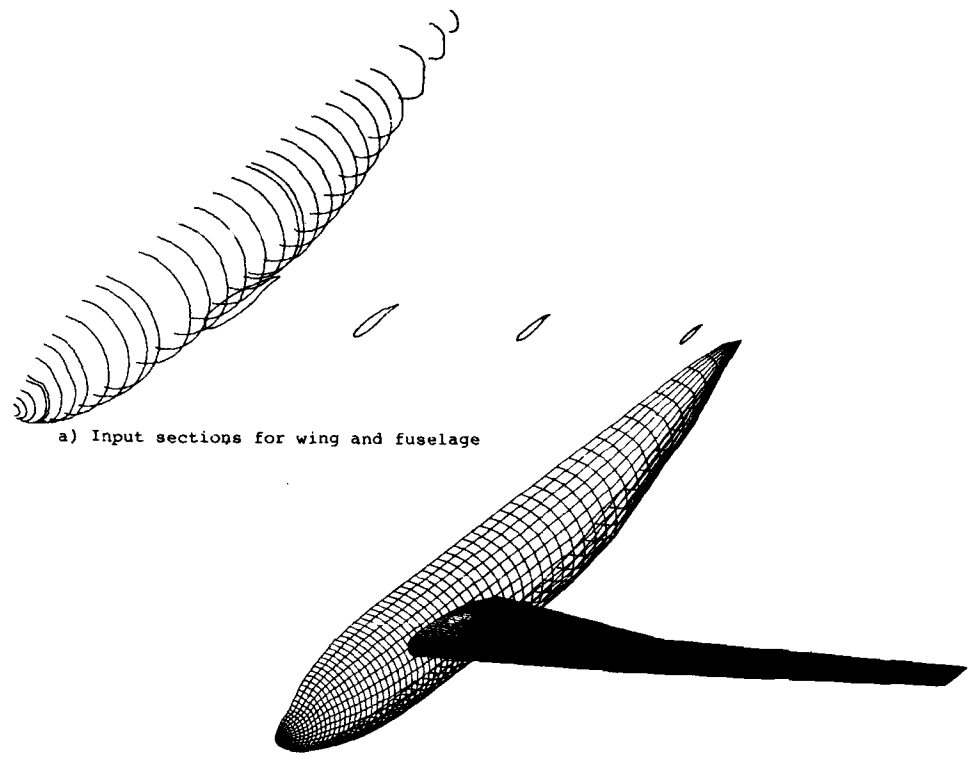
IJKL = 0303

MNOPQR(0) = 002002
MNOPQR(1) = 006005
MNOPQR(2) = 009007
MNOPQR(3) = 014011

b) Definition of segment grid at block boundary i=1

Fig. 6: Definition of block boundaries

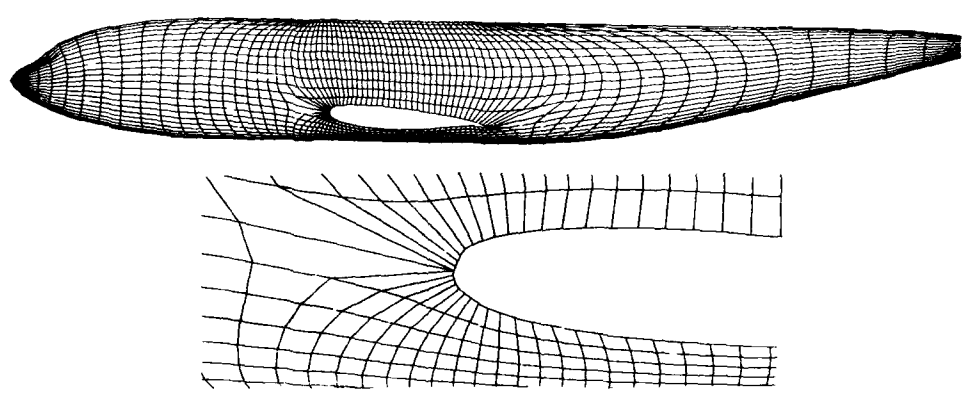
Fig. 7: Definition of index orientation of neighboring segments



a) Input sections for wing and fuselage

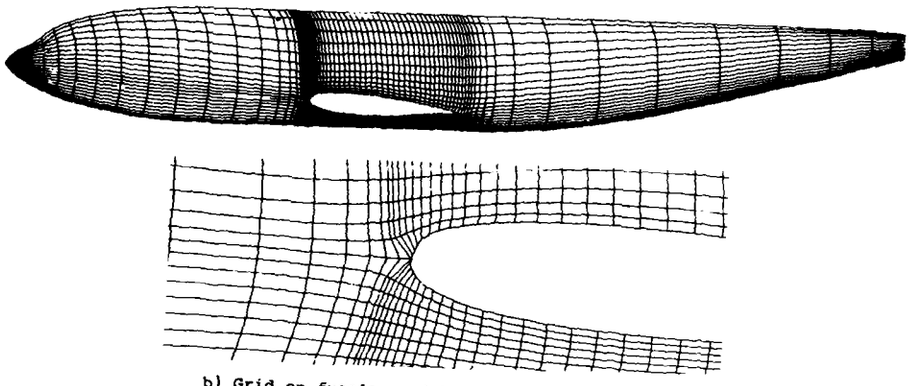
b) Grid of Coons' patches on wing and fuselage

Fig. 8: Surface definition of the DFVLR-F4 Wing-Body

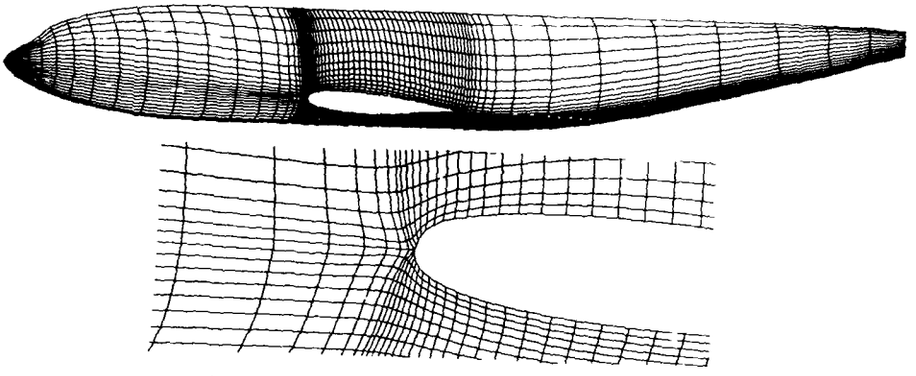


a) Laplace grid on fuselage

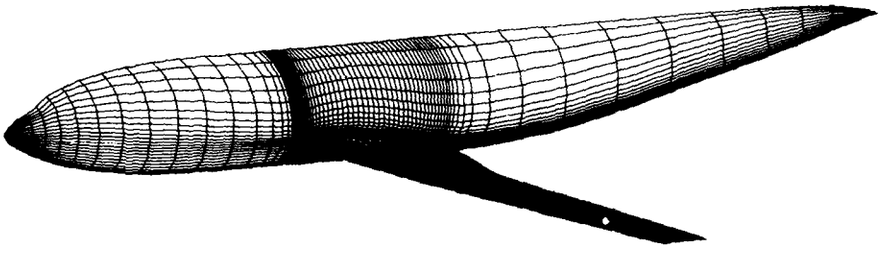
Fig. 9: Surface grid generation for DFVLR-F4 Wing-Body



b) Grid on fuselage with source terms evaluated from equation (6)

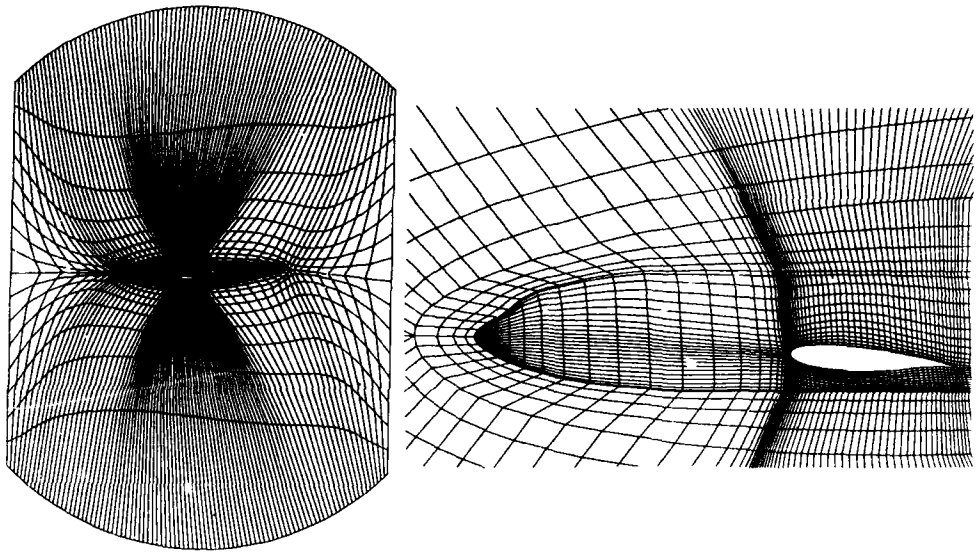


c) Grid on fuselage with iterative grid control using the options INITIAL DISTRIBUTION in streamwise direction and GEOMETRIC PROGRESSION in normal direction

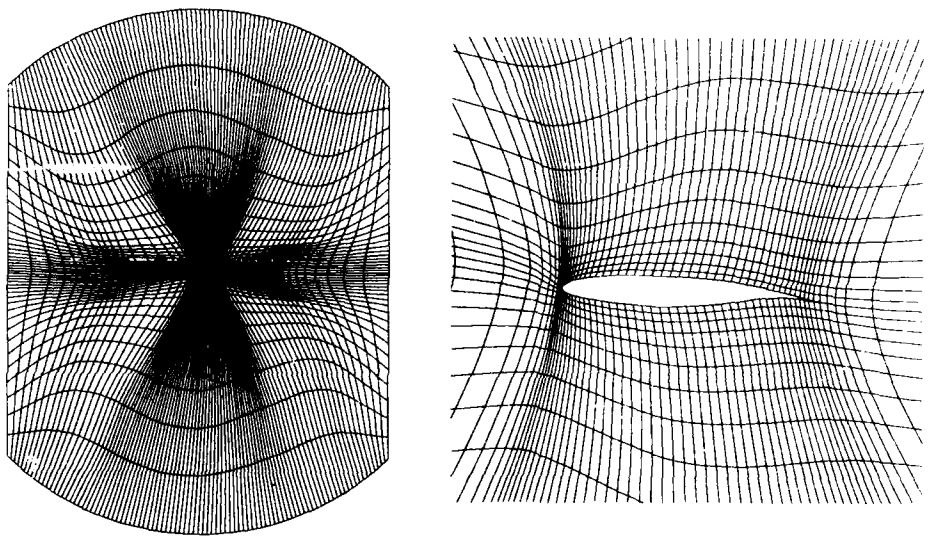


d) Final surface grid of wing and body

Fig. 9: continued

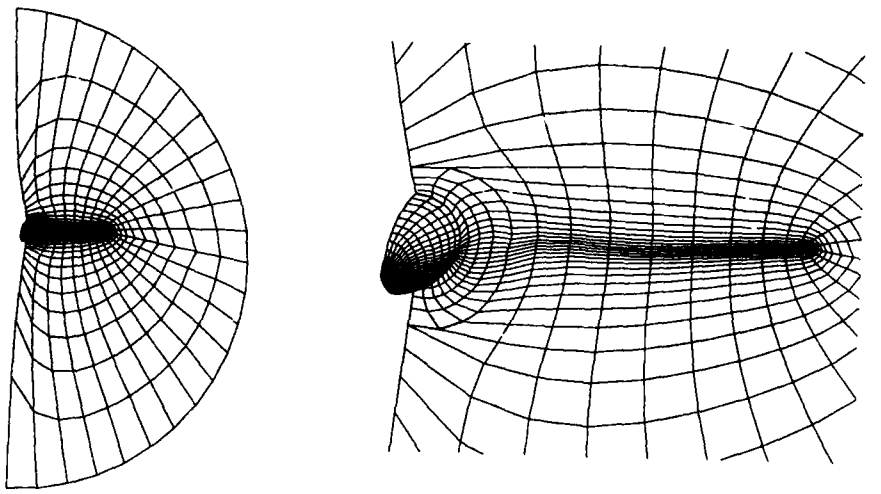


a) Sectionwise grid at the root of the wing

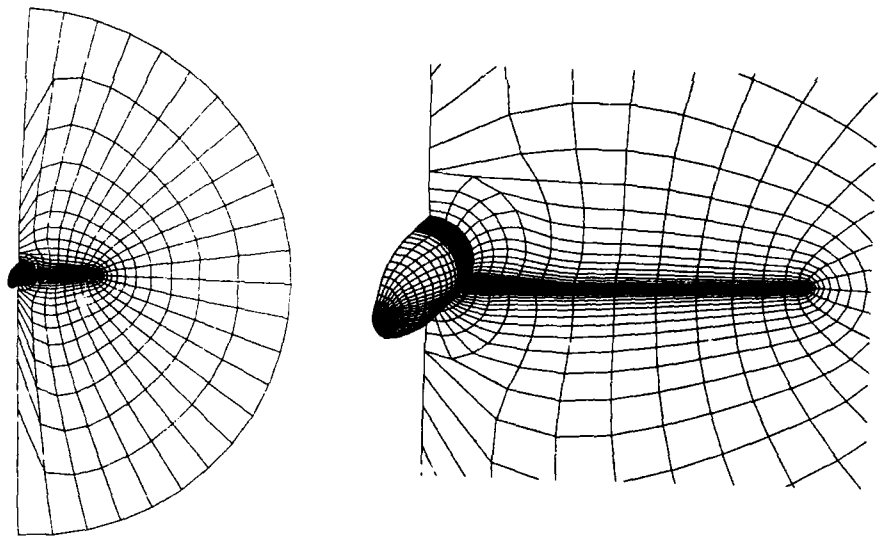


b) Sectionwise grid at the kink of the wing

Fig. 10: Views on field grid around DFVLR-F4 Wing-Body

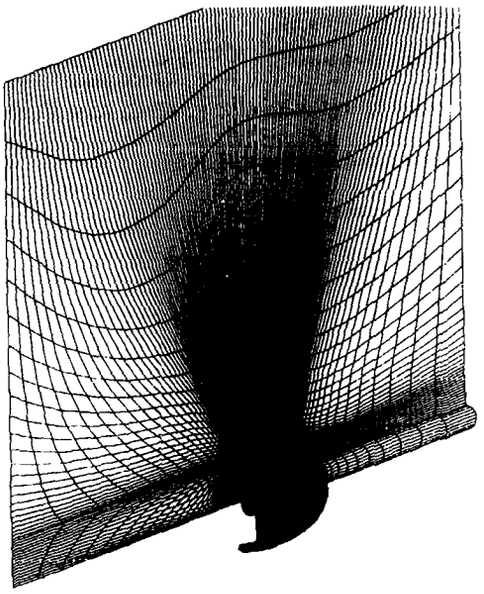


c) Spanwise grid around the fuselage upstream of the wing

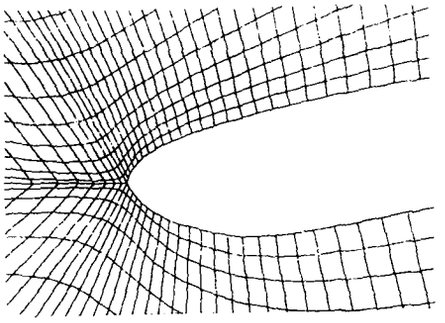


d) Spanwise grid at midchord station of the wing

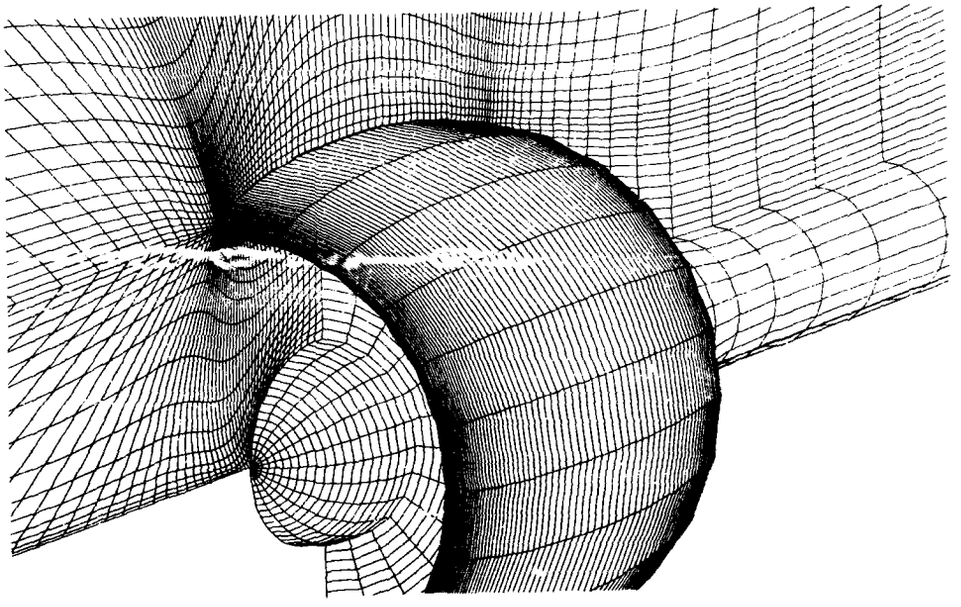
Fig. 10: continued



a) Nacelle and grid in the symmetry plane



c) Grid in the leading edge region with coordinates fixed on the cut upstream of the nacelle



b) Enlarged view of nacelle and grid in the symmetry plane

Fig. 11: Coordinate grid around nacelle

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's Reference	3. Further Reference	4. Security Classification of Document
	AGARD-AG-309	ISBN 92-835-0451-8	UNCLASSIFIED

5. Originator Advisory Group for Aerospace Research and Development
North Atlantic Treaty Organization
7 rue Ancelle, 92200 Neuilly sur Seine, France

6. Title THREE-DIMENSIONAL GRID GENERATION FOR COMPLEX CONFIGURATIONS – RECENT PROGRESS

7. Presented at

8. Author(s)/Editor(s) J.F.Thompson, J.L.Steger
Edited by H.Yoshihara

9. Date March 1988

10. Author's/Editor's Address Various

11. Pages 158

12. Distribution Statement This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.

13. Keywords/Descriptors

Aerodynamic configuration
Grids (coordinates)
Computation
Computer Graphics
Wing Body Configurations
Generating functions
Evaluation

14. Abstract

This AGARDograph surveys some of the capabilities of the CFD community for griding complex three-dimensional configurations. The intent is to provide some insight as to the present state of grid generation for aircraft configurations in order to help assess whether this task presents a long term stumbling block to routine use of CFD in aerodynamic applications. This AGARDograph begins with a brief review of some of the techniques that are available for generating body-conforming curvilinear grids. In order to assess capabilities in grid generation, colleagues at selected institutions were solicited to describe their experiences and difficulties in grid generation of complex configurations. The intent here was not to describe the very latest in grid generation procedures, but to solicit honest comments about what are the difficulties in generating practical grids and what steps are taken to meet these difficulties. These experiences, which comprise the heart of this AGARDograph, are presented as case histories.

Keywords
Computational Fluid Dynamics, Na[illegible]

AGARDograph No.309
Advisory Group for Aerospace Research and Development, NATO
THREE-DIMENSIONAL GRID GENERATION FOR COMPLEX CONFIGURATIONS - RECENT PROGRESS
by J.F.Thompson, J.L.Steger, Edited by H.Yoshihara
Published March 1988
158 pages

This AGARDograph surveys some of the capabilities of the CFD community for griding complex three-dimensional configurations. The intent is to provide some insight as to the present state of grid generation for aircraft configurations in order to help assess whether this task presents a long term stumbling block to routine use of

P.T.O

AGARD-AG-309

Aerodynamic configuration
Grids (coordinates)
Computation
Generating functions
Evaluation

AGARDograph No.309
Advisory Group for Aerospace Research and Development, NATO
THREE-DIMENSIONAL GRID GENERATION FOR COMPLEX CONFIGURATIONS - RECENT PROGRESS
by J.F.Thompson, J.L.Steger, Edited by H.Yoshihara
Published March 1988
158 pages

This AGARDograph surveys some of the capabilities of the CFD community for griding complex three-dimensional configurations. The intent is to provide some insight as to the present state of grid generation for aircraft configurations in order to help assess whether this task presents a long term stumbling block to routine use of

P.T.O

AGARD-AG-309

Aerodynamic configuration
Grids (coordinates)
Computation
Generating functions
Evaluation

AGARDograph No.309
Advisory Group for Aerospace Research and Development, NATO
THREE-DIMENSIONAL GRID GENERATION FOR COMPLEX CONFIGURATIONS - RECENT PROGRESS
by J.F.Thompson, J.L.Steger, Edited by H.Yoshihara
Published March 1988
158 pages

This AGARDograph surveys some of the capabilities of the CFD community for griding complex three-dimensional configurations. The intent is to provide some insight as to the present state of grid generation for aircraft configurations in order to help assess whether this task presents a long term stumbling block to routine use of

P.T.O

AGARD-AG-309

Aerodynamic configuration
Grids (coordinates)
Computation
Generating functions
Evaluation

AGARDograph No.309
Advisory Group for Aerospace Research and Development, NATO
THREE-DIMENSIONAL GRID GENERATION FOR COMPLEX CONFIGURATIONS - RECENT PROGRESS
by J.F.Thompson, J.L.Steger, Edited by H.Yoshihara
Published March 1988
158 pages

This AGARDograph surveys some of the capabilities of the CFD community for griding complex three-dimensional configurations. The intent is to provide some insight as to the present state of grid generation for aircraft configurations in order to help assess whether this task presents a long term stumbling block to routine use of

P.T.O

AGARD-AG-309

Aerodynamic configuration
Grids (coordinates)
Computation
Generating functions
Evaluation

CFD in aerodynamic applications. This AGARDograph begins with a brief review of some of the techniques that are available for generating body-conforming curvilinear grids. In order to assess capabilities in grid generation, colleagues at selected institutions were solicited to describe their experiences and difficulties in grid generation of complex configurations. The intent here was not to describe the very latest in grid generation procedures, but to solicit honest comments about what are the difficulties in generating practical grids and what steps are taken to meet these difficulties. These experiences, which comprise the heart of this AGARDograph, are presented as case histories.

This AGARDograph has been produced at the request of the Fluid Dynamics Panel of AGARD.

ISBN 92-835-0451-8

CFD in aerodynamic applications. This AGARDograph begins with a brief review of some of the techniques that are available for generating body-conforming curvilinear grids. In order to assess capabilities in grid generation, colleagues at selected institutions were solicited to describe their experiences and difficulties in grid generation of complex configurations. The intent here was not to describe the very latest in grid generation procedures, but to solicit honest comments about what are the difficulties in generating practical grids and what steps are taken to meet these difficulties. These experiences, which comprise the heart of this AGARDograph, are presented as case histories.

This AGARDograph has been produced at the request of the Fluid Dynamics Panel of AGARD.

ISBN 92-835-0451-8

CFD in aerodynamic applications. This AGARDograph begins with a brief review of some of the techniques that are available for generating body-conforming curvilinear grids. In order to assess capabilities in grid generation, colleagues at selected institutions were solicited to describe their experiences and difficulties in grid generation of complex configurations. The intent here was not to describe the very latest in grid generation procedures, but to solicit honest comments about what are the difficulties in generating practical grids and what steps are taken to meet these difficulties. These experiences, which comprise the heart of this AGARDograph, are presented as case histories.

This AGARDograph has been produced at the request of the Fluid Dynamics Panel of AGARD.

ISBN 92-835-0451-8